STABILITY AND PERFORMANCE IN PEER TO PEER NETWORKS

BY

JI ZHU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Doctoral Committee:

Professor Bruce Hajek, Chair
Professor Rayadurgam Srikant
Professor Pramod Viswanath
Assistant Professor Philip Godfrey
Assistant Professor Angelia Nedich

Abstract

Stability regions of P2P systems describe analytically the scalability and robustness of the systems; they reveal intrinsic properties of P2P systems and provide intuition and guidance for system design. One contribution of the thesis is to characterize stability regions of BitTorrent-like P2P networks and analyze methods to enlarge the regions. Stability regions are characterized by stochastic models for a variety of popular methods, namely, those providing pieces to peers upon arrival and requiring seeds to dwell. Necessary and sufficient conditions for stabilizing the system are provided and proved. The thesis identifies the least amount of assistance needed from the seeds to stabilize the system. Take the method related to dwelling, for example; the stability can be achieved if every peer dwells on average only long enough to upload one additional piece. Another practical method considered is network coding. It is shown that network coding substantially increases the stability region if a portion of peers arrive with randomly coded pieces; nevertheless, such region remains unchanged if peers arrive without pieces. Lastly, the method of bundling files together, termed as multiswarm networks, is considered, along with the stability region provided. Stability region is shown to be determined by the largest arrival rate of all swarms and insensitive to the number of swarms.

For all methods considered here, stability regions are shown to be identical for various work-conserving piece selection policies. However, these piece selection policies differ significantly in their abilities to prolong the time for an unstable system to stay under normal states. This prompts one to introduce the notion of metastability. In this sense, a piece selection policy prioritizing rarer pieces is advantageous in that it keeps the system metastable far longer.

Another contribution of the thesis is to present an asynchronous distributed algorithm for P2P streaming. The proposed algorithm organizes the streaming topology as multicast trees, one for each substream. The topology

of trees, which is commonly used for streaming, has advantages in low disruption rate, small delay, low interference, good fluency and high efficiency; but it is difficult to build and maintain in a distributed way. The difficulty is tackled by the algorithm, through which cycles are efficiently eliminated and the topology is constantly adjusted towards balanced trees. Compared to existing methods, the algorithm is more robust, converges faster, and scales to as many peers as possible; it does not require any centralized routing servers. Under the algorithm, each peer is guaranteed to be covered by sufficiently many trees with the delay time scaling only logarithmically in the number of peers. The algorithm works under heterogeneous constraints on peer upload capacity, where peers with higher degrees can enjoy lower delay. Thus it fits not only the case of P2P, but also the case of cloud servers.

To my parents, my wife and my child, for their love and support.

Acknowledgments

I am deeply indebted to my advisor Prof. Bruce Hajek for his constant guidance and support at every stage of my PhD studies, without which this dissertation would not be possible. As an excellent researcher and mentor, he taught me how to formulate a problem in the cleanest possible way. His brilliant insights, strict requirements, and aesthetic taste greatly shaped my research. In addition, I appreciate his endless patience and the freedom he gave, which allow me to pursue my research interests that sometimes are well outside the region of P2P networks.

I thank Prof. Rayadurgam Srikant, Prof. Pramod Viswanath, Prof. Philip Godfrey and Prof. Angelia Nedich for being on my doctoral committee and their valuable feedback. I thank Prof. Nitin Vaidya for being on my qualification committee. I am also grateful to Dr. Hao Zhu and Mavis Li for their stimulating suggestions on thesis writing.

I would like to thank the faculty of the Departments of Electrical and Computer Engineering, Mathematics, Computer Science and Industrial and Enterprise Engineering for great curricula and numerous inspirations, especially, Prof. Daniel Liberzon, Prof. Yi Lu, Prof. Jozsef Balogh, Prof. Richard Sowers and Prof. Olgica Milenkovic, for their teaching. I am grateful to Dr. Mudhakar Srivatsa and Dr. Dakshi Agrawal for being my mentors during my internship at IBM Watlabs, where I spent a wonderful summer in 2010. I would also like to acknowledge Dr. Stratis Ioannidis, Dr. Laurent Massoulie and Dr. Hegde Nidhi for stimulating discussions and fruitful collaborations during my internship at Technicolor in 2011.

I have been fortunate to have many friends and colleagues who made my five years of graduate study an unforgettable experience: Xun Gong, Qiaomin Xie, Jiaming Xu, Lili Su, Yunwen Xu, Yun Li, Rui Wu, Tianxiong Ji, Sreekanth Annapureddy, Javad Ghaderi, Aly Elgamal, Jian Ni, Xi Wang, Kun Deng, Rui Ma, Chun Ge, and Jonathan Ligo. In particular, I thank

my roommate Quan Geng for numerous rewarding discussions, during every one of which I gained new knowledge. I thank Dayu Huang, Wei Dai, Sachin Kadloor, Rambo Tan, Vineet Abhishek, Yunwen Xu and Tao Yang for their smart advice in improving presentation skills. My dearest friends in the Crosstalk Association, Zhongzhou Chen, Chen Fu, Longxiang Zhang, Bin Li, Zuanyi Li, Xinyu Zhang, are always there for me during both joyful and stressful times, to whom I will always be thankful.

Finally, it is my greatest honor to thank my family: my mother, my father, my grandparents, my wife and my child. No words could possibly express my deepest gratitude for their love, self-sacrifice and unwavering support. To them I dedicate this dissertation.

Table of Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 BitTorrent-like P2P file distribution

Peer to peer (P2P) networks, due to their ability in scaling demands with service, are widely used in file sharing and streaming. In a P2P network, peers work both as clients to download resources and local servers to contribute. In that way the central server capacity is heavily leveraged. In the past decade, many works have focused on developing efficient algorithms to increase the performance of P2P networks.

The BitTorrent [1] protocol is one of the milestones in designing P2P systems for file sharing. As of February 2013, BitTorrent was responsible for 3.35% of all worldwide bandwidth, more than half of the 6% of total bandwidth dedicated to file sharing [2]. BitTorrent is an asynchronous distributed protocol designed to distribute a very large file to millions of peers. It does not require any central servers, but needs a tracker, whose task is to help new peers find neighbors, working as a key to the entrance of the network. A typical topology of a BitTorrent network is shown in Figure 1.1. The protocol cuts a large file into pieces of fixed size, and further cuts every piece into subpieces for pipelining. Peers holding different subsets of all pieces keep

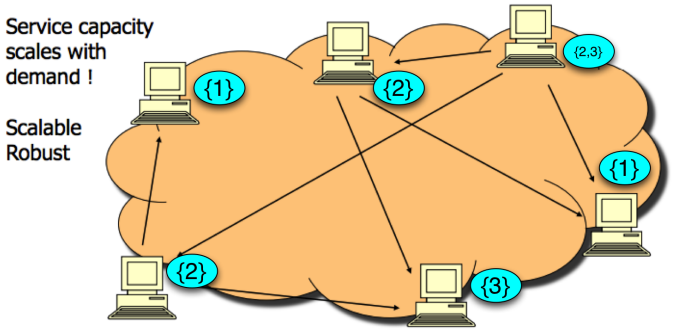


Figure 1.1: Framework of BitTorrent P2P network. $1, 2, 3$ denote pieces.

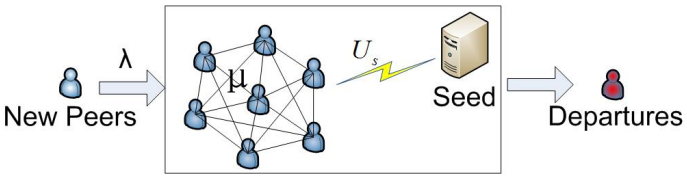


Figure 1.2: Framework of a stochastic model for P2P networks.

constructing links between each other randomly and exchanging pieces. BitTorrent replaces idle links by active ones by the tit-for-tat algorithm so as to guarantee that pieces can flow to all peers. Each peer initially obtains a list of neighbors from the tracker, and always unchokes (uploads to) a fixed number of neighbors, from whom the peer has the highest download rate; periodically each peer randomly unchokes another neighbor, re-ranks the download rates, and chokes (pauses uploading to) neighbors from whom download rates are too low. To maintain diversity of piece distribution, BitTorrent prioritizes transmission of pieces which are rarer in the network. Each peer keeps a local record of the number of neighbors holding each piece, and generally tries to download the rarest piece first.

Various models are proposed to analyze the performance of BitTorrent-like P2P file sharing networks. Stochastic models capturing the piece dissemination process are discussed in [3–8]. Fluid methods analyzing the average case performance are applied in [3, 9–11]. Analysis of scalability is the focus of [3, 4, 6–8, 12]; analysis of throughput is the focus of [5, 10, 13, 14]; the effect of reciprocity and barter is considered in [15]; the effect of proportional replication is considered in [13].

Figure 1.2 shows a widely used stochastic queuing model for BitTorrent-like P2P networks [3–8,12,16,17]. The model assumes new peers arrive according to Poisson processes; seeds and peers periodically sample targets for instantaneous piece downloading (or uploading) at times of Poisson processes as well; and peers depart after collecting all pieces. It naturally captures properties of unstructured random topology, peer and piece selection policies, and heterogeneous upload capacities. Understanding analytically how a BitTorrent-like P2P network behaves over a long period of time is difficult, even under a simple model as that in Figure 1.2. One main difficulty is that the state space is quite large, because the number of possible sets of pieces owned by peers increases exponentially with the number of pieces. Another difficulty is that the piece transmitted from a source peer to a target peer is determined

by sets of pieces owned by the two peers, which are both evolving.

One fundamental question about a BitTorrent-like network is what is its stability region, that is, assuming the seed's upload capacity is U pieces per time unit, what is the largest arrival rate λ of peers that can be supported without the number of peers in the network growing to infinity? Intuitively, as every incoming peer increases the aggregate upload capacity, one would expect BitTorrent to support a very high arrival rate. Determining the stability regions of BitTorrent-like networks can us help to better understand the intrinsic factors affecting the scalability, and thereby provides guidance and intuition for designing P2P networks with stronger robustness and higher performance.

The stability region of BitTorrent-like networks was given recently in [12], which shows that the network is stable if $\lambda < U$, and unstable if $\lambda > U$. This phenomenon is known as the *missing piece syndrome* [12,18], which governed the stability region and undermines the scalability of BitTorrent. Here we continue to formalize the stability regions of BitTorrent-like networks, under a more general model which captures not only all assumptions in [12], but also additional properties like peers dwelling as seeds before departing and new peers obtaining pieces from trackers upon arrival. The findings, which are highly related to the missing piece syndrome, unveil fundamental relations and complete the understanding of robustness of BitTorrent-like networks.

In light of the work on stability regions, a series of works have focused on relieving the missing piece syndrome and extending the stability region [4,11, 16,19]. Approaches proposed in [4,19,20] are to bundle multiple autonomous swarms together into a universal swarm, a method to be introduced in the next section.

1.2 Multiswarm P2P

BitTorrent is shown to be robust and efficient in distributing a very long file to millions of peers [21]. However, it has drawbacks if there are not enough peers simultaneously in the network, either because the file is not long enough or the file is not popular enough. In that case, there may be no paths from seeds to some peers due to topology constraints; the upload capacities provided by peers may not satisfy the download demands. As a

result, a large proportion of peers may suffer from low download rates or be unable to obtain all pieces.

That problem can be relieved by bundling small files together, or bundling unpopular files with popular ones, so as to attract more peers [19]. In simple terms, in a peer to peer network, peers interested in downloading a single file from seed, form a so-called *swarm*. In a multiswarm peer to peer network, multiple files are bundled and cut into pieces, provided by seeds; peers in different swarms aim to obtain different files, but they also spend memory and bandwidth to store and exchange pieces with peers belonging to other swarms.

One benefit of multiswarm networks is the improvement in scalability and robustness, because extra capacity provided by popular swarms can be utilized by other swarms instead of being wasted. Heterogeneous interests of different swarms provide a better diversity in multiswarm networks, where it is more difficult for the missing piece syndrome to occur. It is discovered that bundling swarms together can increase the stability region of BitTorrent-like P2P networks [4], where it is conjectured that a multiswarm network with a fixed seed capacity exhibits an increased stability region compared to autonomous swarms.

Here we formally characterize the stability region of multiswarm P2P networks, proving necessary and sufficient conditions under which multiswarm networks remain stable, under a model similar to that in Figure 1.2. The findings show excellent robustness brought by bundling swarms together, and also reveal tradeoffs between stability and delay.

1.3 P2P streaming by multicast trees

Peer to peer communication is attractive not only for file transferring but also for streaming, due to its advantage in lowering server loads [22–24]. In a peer to peer streaming network, videos (or audios) are cut into pieces and distributed from seeds to peers; peers exchange pieces so as to guarantee that each one can obtain the piece to play before the playback deadline. Scheduling algorithms need to be carefully designed so that upload capacity provided by peers can be utilized efficiently to serve the demand, and each peer can download streams, providing playback continuity with small delay.

The system needs to be robust enough to tolerate peer churn, link failures, congestions, and delays.

Many designs have been proposed for P2P streaming. Unstructured topologies, with peers constantly sampling new targets for new pieces, are considered in [17, 25]. Data dissemination algorithms based on mesh topologies are given in [26,27]. In [28], random Hamiltonian cycles are constructed and tangled and pieces are broadcasted around the union of the cycles. Fixed underlying topologies are considered in [29,30] and flows are scheduled between neighbors. Algorithms to manage multiple distribution trees to disseminate different substreams of a video or audio are discussed in [31–34], but those papers do not concentrate on distributed algorithms.

Unstructured streaming systems are simple to manage and scale well, but playback continuity is sacrificed because constantly building and removing links requires prohibitive overhead. A common design of a P2P streaming network is to cut videos into substreams and apply multicast trees to broadcast different substreams. Tree structures can provide good playback continuity with small startup delay, but can be difficult to manage.

One key issue in designing a P2P streaming network is how to manage multicast trees, which is one topic of the thesis. In contrast to the centralized designs in [31–33], here we focus on distributed algorithms. The model applied includes a complete underlying network for control information so arbitrary peer to peer contact is allowed, like that in Figure 1.2. To model the bandwidth constraint for control information, each peer is only allowed to randomly contact a target from other peers periodically at a certain constant rate, similar to the models applied above for BitTorrent-like P2P networks. This setting, which is more suitable for P2P systems, is different from settings in [35–39], which discuss how to build multicast trees satisfying certain metrics under a fixed underlying topology. Most problems formulated in [35, 37–39] are shown to be NP hard and approximation algorithms are designed. NP hardness is avoided here by assuming a homogeneous and complete underlying topology.

The streaming network is built on top of a complete underlying network. Besides the bandwidth constraint on control information, peers have heterogeneous upload capacities so each of them has a maximum fan-out degree for streaming. Peers have a small buffer to store information about their parents and children in the streaming network. They can exchange messages with

their parents and children, at the same time they can also exchange messages with their sampled targets at the sampling times. As in [31–33], instead of considering a packet level transmission, a flow level model is studied here by assuming that the video is cut into a fixed number of substreams. Source coding like multiple description coding (MDC [40]) can be applied to provide redundancy in substreams. Multiple diverse distribution trees, which constitute the streaming network, are constructed and managed, each for one substream.

In [31] it is mentioned that a good tree management algorithm should maintain 1) short trees, i.e., trees with small depths so as to minimize the probability of disruption due to peer transience or congestion; 2) tree diversity, i.e., the set of ancestors of a peer in each tree are as disjoint as possible so as to increase the effectiveness of the MDC-based distribution scheme; 3) quick processing of peer joins and leaves; and 4) scalability. Centralized solutions in [31–33] are proposed to achieve those goals. One part of the thesis aims to design a distributed tree management algorithm achieving all those goals.

1.4 Main contributions

The thesis is composed of three topics, all applying random sampling models as that in Figure 1.2 and focusing on related questions. The first topic lies in the stability regions of BitTorrent-like P2P networks, the second topic extends the question on stability regions to multiswarm P2P networks, the third topic focuses on designing a distributed multicast tree management algorithm for P2P streaming. The contributions are highlighted below.

In the first topic [7], the basic results of [12] are extended in two particular ways: peers can already have some pieces at the time of their arrival, and peers can dwell for a while in the network after obtaining a complete collection of pieces. The main result of this topic, Theorem 2.1.1, provides the stability region of the network within the space of values of arrival rates, seed and peer uploading capacities, and peer dwelling time. The proof of the main result is shaped by showing that the system either is trapped by the missing piece syndrome, or that it always escapes the missing piece syndrome, depending on the parameter values. Discussions on this topic reveal

the least amount of time peers must dwell after obtaining the entire file so that the whole network is positive recurrent (stable). A corollary of Theorem 2.1.1 is that if each peer can upload one additional piece after obtaining the whole file before departing, the network is stable under any positive seed uploading capacity and any arrival rates. In BitTorrent, the size of a single piece is typically a small fraction of the entire file (about 0.5%) so that it is a light burden for a peer to dwell in the network long enough to upload one more piece after obtaining a complete collection.

Three extensions for Theorem 2.1.1 are also generated in this thesis. The first extension is to point out that Theorem 2.1.1 remains true for a wide variety of piece selection policies, as long as they are work-conserving, i.e., at least one piece is transmitted if the source has useful pieces missed by the target. The second extension is to point out how Theorem 2.1.1 can be modified to incorporate network coding [41, 42]. As a corollary, it is shown that network coding does not increase the stability region if peers arrive without pieces, but if peers arrive with some (randomly coded) pieces, network coding substantially increases the stability region. The third extension is to consider the borderline case, between the necessary and sufficient conditions of Theorem 2.1.1.

In the second topic [8], we formally characterize the stability region of multiswarm P2P networks, proving necessary and sufficient conditions under which such swarms remain stable, summarized as a result in Theorem 3.1.1. It is shown that bundling swarms together significantly increases the stability region, making it scale in the number of bundled swarms. We show that the stability region under limited seed upload capacity is insensitive to the number of swarms, peer uploading capacity and piece selection policies. And we show that the increased stability region of bundled swarms comes at the cost of increased delays, which scale with the total number of bundled swarms. To address the problem of increased delays, a modified system design is proposed which extends the stability region and does not affect delays. Different work-conserving piece selection policies, though lead to the same stability region, are compared on their abilities to resist the missing piece syndrome. It is observed that the multiswarm network becomes metastable when seeds and peers prioritize rare pieces.

Proofs of Theorems 2.1.1 and 3.1.1 are based on the missing piece syndrome and Lyapunov functions. The missing piece syndrome works as a general

framework to determine stability regions of BitTorrent-like P2P networks. Though proofs may be complicated, the missing piece syndrome provides a straightforward idea to qualitatively analyze the scalability of a P2P file sharing network: consider an extreme case where millions of peers are missing the same piece but hold all other pieces in the network, which usually is the worst case. The network is robust if it can recover to normal states from the missing piece syndrome, otherwise it may not be stable. The Lyapunov functions we derived provide a unified stability proof for a broad range of models similar to that in Figure 1.2.

In the third topic [43], an asynchronous distributed algorithm is presented to manage multiple trees for P2P streaming in a flow level model, where videos are cut into substreams. The algorithm has the following properties:

1. Each peer can receive enough substreams.
2. Depths of trees are logarithmic in the number of peers.
3. Trees are diverse and balanced.
4. Cycles are eliminated efficiently in a distributed way.
5. Convergence is fast, providing robustness to peer transience.
6. Peers with higher upload capacity tend to be closer to the roots of the trees.
7. Heterogeneous upload capacity is supported, even in the case a few peers with large degree act as servers and other peers act as clients.
8. Convergence is insured even when the ratio of total demand to total upload capacity, ρ, is one, and it converges more quickly as ρ decreases from one.

A proof of convergence of the algorithm is given assuming instantaneous update of depth information, and for the case of a single tree it is shown that the convergence time is stochastically tightly bounded by a small constant times the log of the number of nodes. These theoretical results are complemented by simulations showing that the algorithm works well even when most assumptions for the theoretical tractability do not hold.

1.5 A brief literature review on P2P networks

This section provides a brief literature review of works related to the thesis, for background about recent research on P2P networks.

1.5.1 File transferring

The first major usage of P2P networks was to distribute large files to millions of nodes at the same time. In this section the BitTorrent application is outlined as an example for discussing previous works on P2P file sharing networks.

The major design aspects of BitTorrent protocol are described in [1], which is introduced in Chapter 2. The paper [9] is among one of the earliest works in analyzing the performance of P2P file sharing networks by modeling. In [9], the BitTorrent working process is separated into two phases, transient regime — which refers to the period of time that a P2P network with a limited number of servers grows its service capacity to meet the load associated with a burst of demands, and steady regime — which refers to the period of time that the service capacity has met the demands. Two different models are applied. In the transient regime, a branching process model is adopted to show that the service capacity can grow exponentially. In the steady regime, a Markov chain model which captures peer churn is provided and numerically computed results are analyzed. At the end of [9], traffic data sampled from a real BitTorrent network is presented to validate the results generated from models. In the branching process model, the authors compare the performance of growth of the P2P network under nonparallel upload — where the file is not cut into pieces and the out degree of each peer is one, parallel uploads — where the file is not cut into pieces and the out degree of each peer is larger than one, and multi-part downloads — where the file is cut into pieces. They show that the network under multi-part downloads possesses a growth rate increased by a factor proportional to the number of pieces, in comparison to the network under nonparallel uploads. They also show that the network under parallel uploads possesses almost no advantage in growth rate, compared to the network under nonparallel uploads. However, the analysis in [9] focuses mainly on the case that files are not cut into pieces. The file dissemination process is ignored, and the effect

of dissemination protocols is not considered.

The work in [10] complements the analysis of P2P file sharing with a two-dimensional fluid model. In the fluid model of [10], the number of downloaders and the number of seeds are taken as the state variables. The authors provide the rest point and the local stability boundary conditions of their fluid model. They analyze the effect of peer selection policies by a noncooperative game. They prove the existence of a Nash-equilibrium of the noncooperative game, and provide a sufficient condition for the system to converge to the equilibrium. At last, experimental results validating the fluid model are provided. To make their analysis tractable, the authors assume pieces are i.i.d. uniformly distributed among all downloaders, which is a reasonable approximation of the piece distribution in a BitTorrent network with a huge number of peers.

While the works in [9, 10] simplify the piece dissemination process to provide closed form estimation of the efficiency, some other work investigates the detailed piece exchanging process for deeper understanding of the throughput.

The paper [5] studies the dissemination of multiple pieces of information from one single source to multiple users. It is assumed that users contact each other randomly, and upload one piece per unit time. It is showed that any one-sided protocols relying only on pushes or only on pulls are inefficient in disseminating all pieces to all users, where one-sided protocols refers to protocols in which piece selection is based on the state of either the transmitter or the receiver, but not both. The authors propose INTERLEAVE — a hybrid one-sided protocol using both pushes and pulls. They show that with high probability, INTERLEAVE can disseminate k pieces to n users in $9(k + \log_2 n)$ time under hard upload constraint, and in $3.2(k + \log_2 n)$ time under soft upload constraint. Efficiency of two-side protocols — which refers to protocols in which piece selection is based on the states of both the transmitter and the receiver — is also investigated in [5]. It is shown that by two-sided protocols it is possible to disseminate n pieces to n users in $n + O(\log_2 n)$ time.

The paper [14] complements the investigation of [5], with different parameters and models. The authors consider the minimal total time (named the minimal makespan) to distribute a P2P file from one server to a fixed number of peers. It is first proved that the minimal makespan is not increased by

an increase in the out degree of all nodes. And the authors show that the minimal makespan is the same as for the simultaneous send/receive version of the broadcasting problem. A systematic centralized piece schedule that achieves the minimal makespan is provided. Moreover, the case of general upload capacities is considered and the authors provide both a mixed integer linear program solution and a fluid limit solution. At last, the authors provide a decentralized way to implement their algorithm, and prove that the throughput of the decentralized algorithm is close to the lower bound.

The papers [5, 14] focus on the efficiency of disseminating pieces to a fixed set of peers, and do not consider peer churn. Some other papers [3, 4, 6–8, 12] which consider the peer churn and study the piece disseminating process focus more on scalability. The works in [7, 8], which are covered in this thesis, apply models similar to that in [5, 14].

The paper [3] captures the detailed piece distribution process with a coupon replication system. In [3], the P2P file transferring network is modeled as a stochastic network, where peers are assumed to arrive with one single piece, and peers apply random peer selection and random useful piece selection policies. Fluid limits of the stochastic model are analyzed in two cases: the layered case, where peers only contact other peers owning the same number of pieces, and the flat case, where peers sample all peers in the network. The paper provides a sufficient condition for the layered system to be globally asymptotically stable, and provides a sufficient condition for the flat system to have a rest point. The paper also investigates the average sojourn time. However, whether the stability of the fluid system implies the stability of the original system is left as an open problem. What is more, to make the fluid model tractable, only symmetric arrival rates and symmetric initial points are considered.

The paper [12] proposes the existence of the missing piece syndrome by a stochastic model which is a composite of models in [3, 5, 10]. The BitTorrent-like P2P network is modeled as a fixed seed distributing a number of pieces to peers, which arrive as a Poisson process and depart as soon as they get all pieces. It is assumed that peers randomly contact each other and upload one piece at a time, similar to the models in [3, 5]. The authors consider a general set of piece selection policies, which all satisfy the constraint that upload cannot be rejected if the uploader has pieces not owned by the downloader. The notion of missing piece syndrome is given, and is shown to be the main

reason for the P2P model's instability. The authors show the stability region, which is determined by the comparison of arrival rate and seed upload rate. The paper also discusses the application of network coding, and provides the stability region. It is shown that network coding does not increase the stability region if peers arrive with empty cache.

The paper in [6] provides a P2P file sharing system aiming to solve the missing piece syndrome. The authors propose an approximate rare-chunk rule on the same model as in [12], but assume that peers sample several neighbors and download only the piece which meets the rare-chunk rule. It is proved that the P2P network under the rare-chunk rule is stable under any arrival rate and seed upload capacity.

While the work of the above papers [3, 6, 12] examines the dissemination of one single file, the paper [4] differs in that it investigates the BitTorrent network in the multiple swarm case. One swarm refers to the group of peers wishing to download the same item. The authors simplify the model by assuming peers arrive with one item in hand, aim to download another item and depart. The stability region of the fluid model is provided. With the fluid model, the authors show that when multiple swarms are combined, only the number of peers in one of the swarms can grow to infinity even if the system is not stable. They provide the fixed point that has the minimum aggregated peer population, and propose an algorithm to achieve the fixed point.

The paper [11] applies an $M/G/\infty$ queue to study the effectiveness of bundling in BitTorrent-like P2P networks. They assume publishers and peers have Poisson arrivals, exponential file downloading time and exponential residence time, and calculate the availability factor, average download time, unserve probability, etc.

The paper [15] investigates the effect of reciprocity and barter in BitTorrent-like P2P network. The authors define a relayless network to be one in which every node transmits only the set of contents that the node initially has, not the contents that the node downloads from other peers. The authors analyze the topology of peer connections, and prove that given a relayless indirect reciprocity network, there is always a corresponding direct reciprocity network such that the loss of efficiency — which is the maximum ratio of node transmission between direct and indirect reciprocity network — is at most two. The benefit of introducing brokers into the BitTorrent-like P2P network

is also simulated in [15].

The paper [13] examines the network-wide benefit of proportional replication for P2P file transferring networks, as well as the distributed protocols to achieve proportional replication. The authors assume that the P2P network topology has exponential expansion, and assume that a fixed number of peers are requesting a fixed number of files with a constant request rate for each file. It is proved that the network bandwidth used is minimized when proportional replication is applied. At last, the paper shows that local storage management algorithms like LRU automatically help to achieve near-proportional replication, and the system performance under LRU is very close to optimal.

Here the discussion on previous works about BitTorrent-like P2P file sharing network is ended. Recently, more researches are focusing on applying P2P networks for streaming. The next section reviews recent papers on live streaming P2P networks.

1.5.2 Live streaming

Live streaming in P2P networks differ from file sharing in that 1) new pieces are constantly generated rather than pre-stored in the seed and 2) piece dissemination has to keep up with the playback deadline. There are many existing live streaming applications, most of them providing not only live streaming service but also video on demand service. In this section, CoolStreaming is outlined as an example of live streaming applications.

The design of CoolStreaming is described in [22]. CoolStreaming originally aims to provide service for broadcasting live football games. Gossiping is the main method applied in CoolStreaming. In CoolStreaming, each peer periodically generates a membership message to announce its existence. The membership message is gossiped to the whole network for other peers to update their peer list. Node failure information is also published through gossiping. CoolStreaming does piece scheduling by first calculating the number of potential suppliers for each piece, then determining the supplier of each piece starting from those pieces with the fewest potential suppliers. Among the multiple potential suppliers for one piece, the one with the highest bandwidth and with enough time to stay in the system is selected. CoolStream is

shown to be scalable and perform well for large numbers of peers.

Various models studying the efficiency of P2P live streaming are proposed. Some models focus on analyzing the flows, like [29, 44, 45]; some models assume pieces are uploaded as a whole instead of as flows, like [25, 28, 46, 47]. These papers all aim to develop reasonable schemes to increase the throughput of live streaming networks.

Papers [46, 47] investigate the method of hybrid piece selection strategies to increase the efficiency of P2P live streaming networks. In [46], the authors propose a discrete stochastic model with one server and a fixed number of peers. It is assumed that the server keeps generating new pieces and keeps distributing new pieces to peers selected uniformly at random. At each time unit, every peer does random peer selection and uploads one piece. To make the model tractable, the authors assume an i.i.d. piece distribution, and assume states of peers to be mutually independent. It is shown that the piece distribution probability under greedy and rarest first piece selection can be expressed by difference equations. Closed form formulas of the piece distribution probability are given by solving the fluid form of the difference equations. The authors show that by mixing the greedy and rarest first strategy, a better performance can be achieved than for either of the two strategies alone. Similar to [46], the paper [47] investigates a piece selection policy which is a hybrid of greedy and rarest first to achieve order optimal performance. The model in [47], assuming the server generates and distributes one new content every time unit, is similar to that in [46]. The authors in [47] show that the greedy and rarest first algorithms have similar buffer scalings (as a function of the number of peers and the target skip-free probability). A hybrid policy which achieves order sense improvements over greedy and rarest first is proposed, and is shown to achieve order optimal performance.

The paper [25] shows that viewing-uploading decoupling channel design — where peers are scheduled to upload pieces they are not viewing — performs much better than isolated channel design — where peers can only upload pieces they are currently viewing. In [25], a queueing network model is considered for multi-channel P2P streaming. The authors assume there are a fixed number of channels and a fixed number of peers, and that peers stay in every channel for exponential time and switch to other channels according to a certain probability. The authors derive the joint distribution of the number of peers viewing different channels, which shows that the decoupling

design performs much better than the isolated design.

The paper [29] proposes an implicit-primal-dual algorithm extended from the primal-dual algorithm to achieve optimal cost for live streaming. The authors model a live streaming network as a single source broadcasting graph, and assign cost to rate at every feasible link. The aim is to minimize the total cost, subject to the min-cut flow being larger than the playback rate. The authors propose the implicit-primal-dual (IPD) algorithm, which takes the number of items in the difference set of two neighbor nodes as the dual variable. The authors show that the global optimal is a fixed point of the fluid model under implicit-primal-dual algorithm, if random linear network coding is applied. At last, numerical evaluations of the system are provided.

The paper [48] studies algorithms to minimize the total server and network delivery cost by placing replica servers closer to various clients. The problem investigated is to route client requests and response streams so as to minimize the total server and network delivery cost. The paper shows that the problem can be solved for example client populations and realistic network topologies. Heuristics are provided for design of practical systems, and it is revealed that the best heuristics can produce systems with total delivery cost within 60% of optimality.

The paper [44] considers a centralized tree construction algorithm to distribute streams with the maximum rate to a fixed number of peers. The authors treat the flow of packets arriving at the server as a stream and classify the stream into multiple substreams, according to the spanning tree which each substream traverses in the network. The authors investigate the problem of maximizing the total streaming rate using trees of substreams rooted as the server. It is shown that the maximum streaming rate can be achieved using $O(logN)$ trees in a network of N peers with homogeneous upload capacities. The authors provide a centralized way to construct trees, which is analogous to the solution of a two-dimensional Block Packing problem.

Centralized algorithms, though attractive in the high throughput provided, are not realistic due to their difficulty in implementation. The authors in [44] propose another paper [28] which complements [44] in that they design a distributed algorithm based on Hamilton cycles for streaming. In [28], the network topology is constructed by superposing multiple random directed Hamilton cycles together. When peers arrive, they break into randomly chosen links in each Hamilton cycle. When peers depart, their parents and

children in each Hamilton cycle re-connect. Peers are required to schedule outgoing links by choosing Hamilton cycles in order. It is shown that the algorithm can achieve the streaming delay of $\log_2 N$ when the streaming rate is less than $1 - \frac{1}{K}$ of the maximum capacity for any integer $K \geq 2$.

The paper [45] investigates the effect of super-peers (peers with high upload capacity) on a live streaming network. The paper applies a stochastic fluid model, in which peers are classified into two classes — super-peer, whose upload capacity is larger than the streaming rate, and ordinary-peer, whose upload capacity is smaller than the streaming rate. Peer churn is considered separately for the two classes as two independent $M/G/\infty$ processes. The authors provide the maximum achievable streaming rate under the fluid flow model. It is shown that when the system is of moderate-to-large size, the system performs well if the ratio of average number of super-peers to average number of ordinary-peers exceeds a certain critical value; otherwise, the system performs poorly.

1.5.3 Video on demand

Video on demand P2P networks differ from live streaming networks in that the resource provided by the former is pre-stored in the server, and peers are allowed to start watching from various parts. In this section some previous work on video on demand is discussed.

PPlive, whose basic design is described in [24], is discussed as an example video on demand network. PPlive cuts movies into chunks, chunks into pieces, and pieces further into subpieces. Movie advertisement and storage is on the chunk level. Movie playback is on the piece level. Transmission is on the subpiece level. PPlive requires every user to offer up to 1GB of storage space, for storing chunks viewed. In PPlive, multiple movies are allowed to be cached at the same time. Users are allowed to upload movies different from the one they are currently watching. When each peer's cache is full, a weight-based evaluation process is taken for chunk-removal. PPlive applies multiple methods for content discovery, including tracker, distributed hash table, and gossiping. A mixed strategy of sequential and rarest-first is applied for piece selection. When transmission happens, a peer requests various contents from multiple neighbors simultaneously, and redirects to other neighbors when the

request time is out. The paper [24] provides measurement, including data on user behavior, user satisfaction, sever load, and health of replication, on PPlive.

Recently, video on demand models can be classified into two categories: content placement models [20,49,50], where videos are distributed from fixed servers (or fixed peers) to peers; or P2P models [51–53], where peers work as servers as well as clients.

As a paper about content placement, the work in [20] proposes a design of P2P video system by decoupling viewing from uploading, in order to decrease the channel switching delay and increase the performance for unpopular channels. Each video is divided into substreams, and peers are divided into substream distribution groups, with each substream distribution group assigned a substream. Peers in the same distribution group are responsible for distributing their corresponding substreams. Groups of substreams of peers do not change though peers switch channels, but may adapt slowly to meet the evolving channel popularity. An adaptive algorithm is proposed to assign peers to substreams, and simulation is provided to validate its effectiveness.

The paper [49] studies a multi-video P2P VOD system using a two-dimensional fluid model, with states being the number of downloaders and the number of seeds, similar to [10]. The authors assume peers watching each video arrive as a Poisson process, and separate the departures into two types — 1) peers depart at a constant rate, before the playback ends; 2) peers depart after watching ends. The model is linearized and corresponding performance results are evaluated.

A static model of VOD is studied in [50]. In the model, n set-top boxes (servers) with identical upload and storage capacities collaborate to serve r videos simultaneously. The upper and lower bounds on the catalog size are given, where the catalog size is defined as the maximal number of distinct videos that can be stored in the server so that any demand of at most r videos can be served. It is shown that the catalog size is constrained by the storage capacity, the upload capacity, and the maximum out degree of a server, as well as r/n. The paper shows that the achievable catalog size drastically increases when the upload capacity of the servers becomes strictly greater than the video playback rate.

As a paper about P2P models, the work in [51] studies the effectiveness

of movie replication algorithms by modeling the VOD network as a static stochastic queue. It is assumed that each peer requesting a piece can download from all peers owning the piece. And the uplink bandwidth of each uploader is equally divided to serve all outstanding requests. The model considers two user behaviors, deterministic behavior and random jump behavior, with the latter denoted by a transition matrix. A random load balancing algorithm is proposed for piece distribution, and is compared through simulation with other algorithms.

The paper [52] studies video on demand networks in both a content placement model and a P2P model. In the content placement model, the paper assumes there is a fixed number of servers (boxes) in the network, and that requests for every specific item arrive as independent Poisson processes. A proportional-to-product placement strategy is proposed and it is shown that it can achieve the optimal acceptance rate asymptotically as the number of servers goes to infinity. In the P2P model, the paper assumes there is a fixed number of peers, and that requests for specific contents originate from a peer chosen uniformly at random from among all peers. A hot-warm-cold strategy is proposed and is shown to be optimal asymptotically as the number of peers goes to infinite.

The paper [53] proposes a video on demand system named Push-to-Peer, where content is proactively pushed to peers, and persistently stored before the actual P2P transfers. The system has two phases — push phase, when the content server pushes videos to peers, and pull phase, when peers retrieve missing content from other peers. The paper investigates full-stripping data placement scheme and code-based data placement scheme, and proves optimality properties, in terms of deterministic demands and stochastic demands.

1.6 Thesis organization

The thesis is organized as follows: In Chapter 2, we discuss the stability region of BitTorrent-like P2P networks, which is related to the work in [7]; discussions on the extended stability region of multiswarm P2P networks, related to the work in [8], is provided in Chapter 3; in Chapter 4, a distributed algorithm to manage multicast trees for P2P streaming is presented, which is related to the work in [43]; the thesis concludes in Chapter 5.

Chapter 2

SINGLE SWARM FILE DISTRIBUTION

2.1 Model and the stability region

The model is a merger of models in [3, 9, 10]. It captures random sampling, peer arrivals and departures, homogeneous capacity constraint, and work-conserving piece selection policies.

In a P2P network, a fixed seed owning a large file is always present. The large file is divided into K pieces, for some $K \geq 1$. Label the pieces as $1, 2, ...K$ and apply $\mathcal{F}$ to denote the set of all pieces. The goal is to distribute the large file, piece by piece, from the seed to all peers, which arrive to the network at different times and depart once they collect the file. At any time, each peer in the network holds some subset of $\mathcal{F}$. For any subset C of $\mathcal{F}$, a peer holding the collection of pieces C is called a *type C peer*. A type $\mathcal{F}$ peer is called a *peer seed*. To capture the case that peers can download some pieces from trackers upon their arrival, assume type C peers arrive to the network at times of a Poisson process with rate λ_C. Although all possible values of $(\lambda_C, C \subseteq \mathcal{F})$ are considered in the model, typically in practice λ_C is small or equal to zero when $|C| > 1$.

A complete underlying network is considered, so arbitrary node to node contact and piece exchange is permitted. To model the bandwidth constraint, we only allow each node, either the fixed seed or a peer, to randomly contact and upload at most one piece to a target selected from other nodes periodically at a certain constant rate. The contact rate is homogeneous and denoted by μ for all peers, and denoted by U for the fixed seed. Specifically, assume the fixed seed and each peer maintain internal Poisson clocks; the clock of the fixed seed ticks at rate U, and the clock of any peer ticks at rate μ. Whenever the clock of the fixed seed ticks, the fixed seed contacts a peer as a target, which is selected uniformly from among all peers. The fixed seed

checks to see whether the target needs any pieces, and uploads to the target the copy of one piece uniformly selected from among the pieces it needs. If the target does not need any pieces (because the target is a peer seed), no piece is uploaded and the fixed seed remains silent between clock ticks.

A peer similarly uploads pieces. When its rate μ Poisson clock ticks, it contacts a target selected uniformly at random from all peers, and checks to see whether it has pieces needed by the target. If the answer is yes, it uploads to the target a copy of a piece uniformly chosen from among its pieces needed by the target; if the answer is no, no piece is uploaded and the peer does not upload pieces between clock ticks.

Notice that for theoretical tractability the peer contacts and piece uploads are assumed to be instantaneous. That assumption is reasonable if the time for peer contact and piece uploading is shorter than the time between clock ticks. At most one piece is uploaded upon each clock ticking, so $1/\mu$ and $1/U$ are approximately the average piece transmission time from peer to peer and from seed to peer in a P2P network. To summarize, it is assumed that the fixed seed and all peers apply the *random peer contact* and *random novel piece upload* strategies at instants of Poisson processes, with the contact-upload rate of the fixed seed denoted by U and the contact-upload rate of each peer denoted by μ.

Assume that each peer, after becoming a peer seed, dwells in the system for an exponentially distributed length of time with mean $1/\gamma$, with $0 < \gamma \leq \infty$. The case $\gamma = \infty$ is shorthand notation for the case that peers depart immediately after collecting all pieces. Intuitively, smaller values of γ yield better system performance, because peer seeds can upload more pieces if they stay in the system longer. Theorem 2.1.1 identifies the smallest mean peer seed dwelling time (i.e. largest γ) sufficient for a stable system. If the rate U of the fixed seed is sufficiently large, or if the rates λ_C are large enough for some nonempty C, the system can be stable even if peers do not become peer seeds (i.e. even if $\gamma = \infty$). The arrivals of peers, the peer seed dwell times, and the ticking of Poisson clocks, are mutually independent. Notations and assumptions are summarized as follows:

- $\mathcal{C}$: Set of all subsets of $\mathcal{F} = \{1, \ldots, K\}$, where $K \geq 1$ is the number of pieces, and $\mathcal{F}$ is the collection of all pieces.

- Type C peer: A peer with set of pieces $C \in \mathcal{C}$ is a type C peer, which

becomes a type $C \cup \{i\}$ peer if the seed or another peer uploads piece $i \notin C$ to it. A type $\mathcal{F}$ peer is also called a peer seed.

- Type C group: The set of all type C peers in the system.
- Arrivals: Exogenous arrivals of type C peers form a rate $\lambda_C \in [0, \infty)$ Poisson process. To avoid triviality, assume the total arrival rate of peers — $\lambda_{total} = \sum_{C:C\in\mathcal{C}} \lambda_C$ — is strictly positive. Also, without loss of generality, if $\gamma = \infty$, assume $\lambda_{\mathcal{F}} = 0$.
- Random peer contact: The fixed seed contacts a uniformly chosen peer at instants of a Poisson process with rate $U \in [0, \infty)$. Every peer contacts a uniformly chosen peer at instants of a Poisson process with rate $\mu \in (0, \infty)$.
- Random novel piece upload: When A contacts B, if B does not have all pieces that A has, A uploads to B a copy of one piece uniformly chosen from among the pieces A has but B does not have. Otherwise no piece is uploaded.
- Departures: If $\gamma \in (0, \infty)$, every peer becomes a peer seed after obtaining all K pieces, and subsequently remains in the system for an exponentially distributed length of time with mean $1/\gamma$ before departing. If $\gamma = \infty$, then $\lambda_{\mathcal{F}} = 0$ and peers depart immediately after obtaining all K pieces.

Under the assumptions above, the system is a Markov chain with state vector $\mathbf{n} = (n_C : C \in \mathcal{C}) \in \mathbb{N}^{|\mathcal{C}|}$ if $\gamma \in (0, \infty)$, and $\mathbf{n} = (n_C : C \in \mathcal{C} \setminus \{\mathcal{F}\}) \in \mathbb{N}^{|\mathcal{C}|-1}$ if $\gamma = \infty$, where n_C is defined to be the number of type C peers, except we define $n_C = 0$ in the case $C = \mathcal{F}$ and $\gamma = \infty$. Define $\Gamma_{C,C'}$ for $C, C' \in \mathcal{C}$ as follows:

$$\Gamma_{C,C'} := \begin{cases} \frac{n_C}{n}\left(\frac{U}{|\mathcal{F}\setminus C|} + \sum_{S:i\in S\in\mathcal{C}} \frac{n_S\mu}{|S\setminus C|}\right) & \text{if } i \in \mathcal{F}\setminus C, C' = C\cup\{i\}, n \geq 1 \\ 0 & \text{else} \end{cases} \tag{2.1}$$

where $n := \sum_{C:C\in\mathcal{C}} n_C$ is the total number of peers. In words, unless $C' = \mathcal{F}$ and $\gamma = \infty$, $\Gamma_{C,C'}$ is the aggregate rate of transition of peers from type C to type C'; If $C' = \mathcal{F}$ and $\gamma = \infty$, $\Gamma_{C,C'}$ is the aggregate rate of departures from the system of peers of type C.

Let $\mathbf{e}_C$ denote the vector with the same dimension as $\mathbf{n}$, with a one in coordinates C and other coordinates equal to zero. The positive entries of the generator matrix $Q = (q(\mathbf{n}, \mathbf{n}'))$ are given by:

- if $\gamma \in (0, \infty)$, $\mathbf{n} = (n_C : C \in \mathcal{C})$,

$$\begin{aligned} q(\mathbf{n}, \mathbf{n} + \mathbf{e}_C) &= \lambda_C \\ q(\mathbf{n}, \mathbf{n} - \mathbf{e}_{\mathcal{F}}) &= \gamma n_{\mathcal{F}} \\ q(\mathbf{n}, \mathbf{n} - \mathbf{e}_C + \mathbf{e}_{C\cup\{i\}}) &= \Gamma_{C,C\cup\{i\}}, \quad \text{if } C \subsetneq \mathcal{F}, i \notin C. \end{aligned}$$

- if $\gamma = \infty$, $\mathbf{n} = (n_C : C \in \mathcal{C} \setminus \{\mathcal{F}\})$,

$$\begin{aligned} q(\mathbf{n}, \mathbf{n} + \mathbf{e}_C) &= \lambda_C \\ q(\mathbf{n}, \mathbf{n} - \mathbf{e}_C + \mathbf{e}_{C\cup\{i\}}) &= \Gamma_{C,C\cup\{i\}}, \quad \text{if } C \in \mathcal{C}, |C| \leq K-2, i \notin C. \\ q(\mathbf{n}, \mathbf{n} - \mathbf{e}_C) &= \Gamma_{C,\mathcal{F}}, \quad \text{if } C \in \mathcal{C}, |C| = K-1. \end{aligned}$$

The following definitions of stability and instability for the Markov process [54] is applied to describe the stability region.

Definition 2.1.1 *The system is* unstable *if it is transient and the number of peers converges to infinity with probability one; and the system is* stable *if it is positive recurrent and it has a finite mean number of peers in equilibrium.*

Theorem 2.1.1 describes the stability region of the P2P system.

Theorem 2.1.1 *(a). Given $U \in [0, \infty)$, $\mu \in (0, \infty)$, $\gamma \in (0, \infty]$, $\{\lambda_C : C \in \mathcal{C}, \lambda_C \in [0, \infty)\}$ with $\lambda_{\mathcal{F}} = 0$ if $\gamma = \infty$, and $\lambda_{total} > 0$, the Markov process with generator matrix Q is transient (unstable) if either of the following two conditions is true:*

- *$0 < \mu < \gamma \leq \infty$ and for some $k \in \mathcal{F}$,*

$$\lambda_{total} > \left[U + \sum_{C: k\in C, C\in\mathcal{C}} \lambda_C (K + 1 - |C|) \right] \left(\frac{1}{1 - \mu/\gamma} \right) \tag{2.2}$$

- *$0 < \gamma \leq \mu$ and for some piece $k \in \mathcal{F}$, no copies of piece k can enter the system.*

(b). Conversely, the process is positive recurrent and $E[n] < \infty$ in equilibrium (stable), if either of the following two conditions is true:

- $0 < \mu < \gamma \leq \infty$ *and for any* $k \in \mathcal{F}$,

$$\lambda_{total} < \left[U + \sum_{C:k\in C, C\in\mathcal{C}} \lambda_C (K + 1 - |C|) \right] \left(\frac{1}{1 - \mu/\gamma} \right) \tag{2.3}$$

- $0 < \gamma \leq \mu$ *and for any* $k \in \mathcal{F}$, *it is possible for new copies of piece* k *to enter the system.*

We remark that when we say new copies of piece k can enter the system, we mean $U > 0$ or $\lambda_C > 0$ for some $k \in C \in \mathcal{C}$. And we remark that condition (2.3) holding for all $k \in \mathcal{F}$ is equivalent to the following: for any $S \in \mathcal{C} \setminus \{\mathcal{F}\}$,

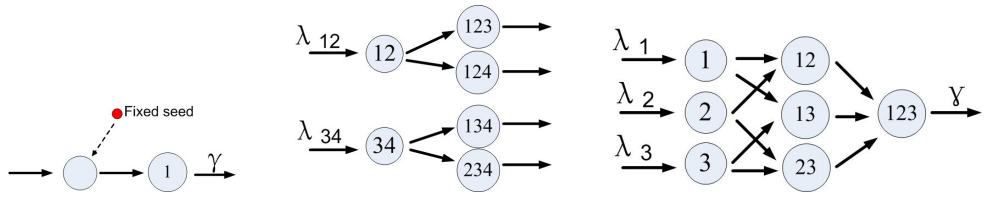
$$\begin{aligned} \triangle_S &:= \lambda_{total} - \frac{U + \sum_{C:k\in C, C\in\mathcal{C}} \lambda_C (K + 1 - |C|)}{1 - \mu/\gamma} \\ &= \sum_{C:C\subseteq S} \lambda_C - \frac{U + \sum_{C:C\not\subseteq S} \lambda_C \left(K - |C| + \mu/\gamma\right)}{1 - \mu/\gamma} < 0. \end{aligned} \tag{2.4}$$

In particular, (2.4) holds for all $S \in \mathcal{C} \setminus \{\mathcal{F}\}$ if and only if it holds for all $S \in \{\mathcal{F} \setminus \{i\} : i \in \mathcal{F}\}$.

2.2 Examples illustrating the stability region

To illustrate Theorem 2.1.1, we describe stability regions of three examples shown in Figure 2.1. Each node in Figure 2.1 denotes a group of peers with the same type. The arrows in Figure 2.1 illustrate the flow of peers from one type to another. For example, node 12 in Figure 2.1(b) denotes the group of peers holding the set of pieces $\{1, 2\}$; the arrow marked λ_{12} in Figure 2.1(b) denotes that the arrival rate of peers holding $\{1, 2\}$ is λ_{12}.

Example 1: This example is treated in [55]. In Figure 2.1(a), the file is transferred as a single piece, that is, $K = 1$. New peers without any piece arrive into the system at the times of a Poisson process with rate λ_0. After obtaining the piece a peer becomes a peer seed. At rate U, the fixed seed contacts and uploads the piece to new peers, who become peer seeds



(a) $K = 1, \gamma < \infty$ (b) $K = 4, \gamma = \infty$ (c) $K = 3, \gamma < \infty$

Figure 2.1: Flows of peers in three example P2P networks.

after obtaining the piece. When peer seeds are in the system, they randomly contact and upload copies of the piece to new peers with rate μ, which creates more peer seeds. After staying for an exponentially distributed time period with mean $1/\gamma$, a peer seed leaves the system. This example illustrates our model with parameters $K = 1$, $U, \mu, \gamma, \lambda_\emptyset = \lambda_0 \in (0, \infty)$, and $\lambda_{\{1\}} = 0$.

The stability of a system is determined by its ability to recover from a heavy load. First consider the case that there are many peer seeds in the system. Because every peer seed departs at rate γ, in essence, the service rate $\gamma n_{\mathcal{F}}$ scales linearly with the number of peer seeds, $n_{\mathcal{F}}$, as in an infinite server system, so the system can recover no matter how many peer seeds there are. Secondly consider the case that there are millions of type $\emptyset$ peers and several peer seeds, such that the fraction of type $\emptyset$ peers is close to one. For a long time period, when the fixed seed or a peer seed randomly contacts a peer to upload a piece, the probability they contact a type $\emptyset$ peer is close to one. So the group of type $\emptyset$ peers receives uploads from the fixed seed at rate almost U. Once a peer becomes a peer seed, it can upload more pieces to type $\emptyset$ peers, creating more peer seeds, which upload more pieces. So every peer seed can create a branching process [56] of departures from the group of type $\emptyset$ peers. The mean time for a peer seed to stay in the system is $1/\gamma$, and during its stay it uploads pieces to type $\emptyset$ peers at rate close to μ. So on average, each peer seed can upload to approximately μ/γ type $\emptyset$ peers. By the theory of branching process, if $\mu/\gamma \geq 1$, the expected number of descendants of a peer seed is infinite, which stabilizes the process because almost all type $\emptyset$ peers may become the descendants of the peer seed. If $\mu/\gamma < 1$, on average every peer seed has $\frac{\mu/\gamma}{1-\mu/\gamma}$ descendants. Hence, every upload of the piece from the fixed seed to a type $\emptyset$ peer causes, on average, about $\frac{1}{1-\mu/\gamma}$ departures from the type $\emptyset$ group. Comparing to λ_0, the arrival rate of type $\emptyset$ peers, this suggests that the system is stable if either $\mu \geq \gamma$,

or $\mu < \gamma$ and $\lambda_0 < U\frac{1}{1-\mu/\gamma}$. Conversely, if $\mu > \gamma$ and $\lambda_0 > U\frac{1}{1-\mu/\gamma}$, the arrival rate of type $\emptyset$ peers is larger than the average rate of departures from the type $\emptyset$ group, indicating that the system cannot always recover from the heavy load of type $\emptyset$ group and so it is unstable. This conclusion is confirmed by [55] and Theorem 2.1.1.

Example 2: As shown in Figure 2.1(b), the file is divided into four pieces, that is, $K = 4$. There are two types of new peers arriving, type $\{1, 2\}$ and type $\{3, 4\}$, which arrive as two independent Poisson processes with respective rates λ_{12} and λ_{34}. There is no fixed seed in the system. Peers contact and upload pieces to each other so that they can depart. Peers depart immediately after obtaining all four pieces, that is, $\gamma = \infty$; there are no peer seeds in the system. This example illustrates our model with parameters $K = 4$, $U = 0$, $\gamma = \infty$, $\mu, \lambda_{\{1,2\}} = \lambda_{12}, \lambda_{\{3,4\}} = \lambda_{34} \in (0, \infty)$, $\lambda_C = 0$ for $C \neq \{1, 2\}, \{3, 4\}$.

Consider the ability of the system to recover from a heavy load. First, consider the network starting from a state such that all peers are type $\{1, 2, 4\}$ and there are so many type $\{1, 2, 4\}$ peers that the fraction of them among all peers is close to one for a long time. On one hand, most new type $\{1, 2\}$ peers download piece 4 from a type $\{1, 2, 4\}$ peer and join the type $\{1, 2, 4\}$ group, so the arrival rate of type $\{1, 2, 4\}$ peers is close to λ_{12}. On the other hand, most new type $\{3, 4\}$ peers download pieces 1 and 2 from type $\{1, 2, 4\}$ peers and then depart, with an expected lifetime in the system approximately $\frac{2}{\mu}$. During its lifetime, a type $\{3, 4\}$ peer uploads piece 3 to two type $\{1, 2, 4\}$ peers on average and thereby induces two departures on average. So the medium term aggregate departure rate of type $\{1, 2, 4\}$ peers is close to $2\lambda_{34}$. Hence, if $\lambda_{12} < 2\lambda_{34}$, the system is able to recover from a heavy load of type $\{1, 2, 4\}$ (or $\{1, 2, 3\}$) peers. Conversely, if the inequality goes the other way, that is, $\lambda_{12} > 2\lambda_{34}$, the arrival rate of type $\{1, 2, 4\}$ peers is larger than the aggregate departure rate of type $\{1, 2, 4\}$ peers. So the type $\{1, 2, 4\}$ group will keep growing. Thus if $\lambda_{12} > 2\lambda_{34}$ the system cannot always recover from a heavy load of type $\{1, 2, 4\}$ (or $\{1, 2, 3\}$) peers. Similarly, if $\lambda_{34} < 2\lambda_{12}$ the system can recover from a heavy load of type $\{2, 3, 4\}$ (or $\{1, 3, 4\}$) peers. And the system cannot always recover from the same heavy load if $\lambda_{34} > 2\lambda_{12}$.

The situation is similar if there is a heavy load of type $\{1, 2\}$ (or $\{3, 4\}$) peers, while there are few peers with other types. The arrival rate of type $\{1, 2\}$ peers is λ_{12}. The aggregate departure rate of type $\{1, 2\}$ peers, from

both the uploads from new arrived type $\{3,4\}$ peers and from type $\{1,2,x\}, x = 3,4$ peers (which are formerly type $\{1,2\}$ peers), is larger than $2\lambda_{34}$. So if $\lambda_{12} < 2\lambda_{34}$ the system is able to recover from the heavy load of type $\{1,2\}$ peers.

Secondly, consider the case that there are heavy loads in groups of at least two types, e.g. type $\{1,2\}$ and $\{1,2,3\}$. There is at least one type of peer that can upload to the other type of peer, e.g. type $\{1,2,3\}$ peers can upload to type $\{1,2\}$ peers. There are many uploads from type $\{1,2,3\}$ peers to type $\{1,2\}$ peers so that the departure rate from type $\{1,2\}$ group is large, which stabilizes the system. This suggests that the system is stable if $\lambda_{12} < 2\lambda_{34}$ and $\lambda_{34} < 2\lambda_{12}$, and unstable if either $\lambda_{12} > 2\lambda_{34}$ or $\lambda_{34} > 2\lambda_{12}$. This conclusion is confirmed by Theorem 2.1.1.

Example 3: As shown in Figure 2.1(c), the file is divided into three pieces, that is, $K = 3$. New peers arrive at a total rate λ_{total}, and each peer arrives with one piece, having piece i with probability $\lambda_i/\lambda_{total}$. So there are three types of new peers, type $\{1\}$, type $\{2\}$, and type $\{3\}$, which arrive as three independent Poisson processes with rates λ_1, λ_2 and λ_3, respectively. There is no fixed seed in the system. At rate μ each, peers randomly contact and upload pieces to each other. After collecting all three pieces, every peer stays in the system as a peer seed for an exponentially distributed time with mean $1/\gamma, \gamma > \mu$. This example illustrates our model with parameters $K = 3$, $U = 0$, $0 < \mu < \gamma \leq \infty$, $\lambda_{\{1\}} = \lambda_1, \lambda_{\{2\}} = \lambda_2, \lambda_{\{3\}} = \lambda_3 \in (0,\infty)$, $\lambda_C = 0$ for $|C| \neq 1$.

Consider whether the system can recover from a heavy load. First, consider the network starting from a state such that all peers are type $\{1,2\}$ and there are so many type $\{1,2\}$ peers that the fraction of them among all peers is close to one for a long time. Almost every new type $\{1\}$ and type $\{2\}$ peer joins the type $\{1,2\}$ group because it can download as a high rate from peers with type $\{1,2\}$. So the arrival rate of the type $\{1,2\}$ group is close to $\lambda_1 + \lambda_2$. Over the medium term, every new type $\{3\}$ peer has an expected lifetime approximately $\frac{2}{\mu} + \frac{1}{\gamma}$, with $\frac{2}{\mu}$ being the expected time for the type $\{3\}$ peer to download two pieces from type $\{1,2\}$ peers, and with $\frac{1}{\gamma}$ being the expected time for the type $\{3\}$ peer to be a peer seed. During its lifetime every type $\{3\}$ peer uploads approximately $2+\frac{\mu}{\gamma}$ pieces to type $\{1,2\}$ peers on average. By the reasoning of example one, every peer seed creates a branching process of departures of type $\{1,2\}$ peers, with the total number of new peer

seeds (including the root) equal to $\frac{1}{1-\mu/\gamma}$. Thus, on average, every new type $\{3\}$ peer induces $(2+\frac{\mu}{\gamma})\frac{1}{1-\mu/\gamma}$ departures from type $\{1,2\}$ group, so the medium term aggregate departure rate of type $\{1,2\}$ peers is approximately $\lambda_3(2+\frac{\mu}{\gamma})\frac{1}{1-\mu/\gamma}$. Hence if $\lambda_1+\lambda_2 < \lambda_3(2+\frac{\mu}{\gamma})\frac{1}{1-\mu/\gamma}$, the system is able to recover from a heavy load of type $\{1,2\}$ group. Conversely, if $\lambda_1+\lambda_2 > \lambda_3(2+\frac{\mu}{\gamma})\frac{1}{1-\mu/\gamma}$, type $\{1,2\}$ group will keep increasing and the system cannot always recover from the heavy load. Similarly, if $\lambda_2+\lambda_3 < \lambda_1(2+\frac{\mu}{\gamma})\frac{1}{1-\mu/\gamma}$, or $\lambda_1+\lambda_3 < \lambda_2(2+\frac{\mu}{\gamma})\frac{1}{1-\mu/\gamma}$, the system is able to recover from a heavy load of type $\{2,3\}$, or $\{1,3\}$ group. And if either of the two inequalities is reversed, the system cannot always recover from a corresponding heavy load.

Secondly, through considerations similar to those in example one and two, we can see that the conditions of heavy load in other single-type group or heavy load in multiple-type groups can also be recovered from if the three inequalities above hold. This suggests that the system is stable if

$$\begin{cases} \lambda_1+\lambda_2 < \lambda_3(2+\frac{\mu}{\gamma})\frac{1}{1-\mu/\gamma} \\ \lambda_2+\lambda_3 < \lambda_1(2+\frac{\mu}{\gamma})\frac{1}{1-\mu/\gamma} \\ \lambda_1+\lambda_3 < \lambda_2(2+\frac{\mu}{\gamma})\frac{1}{1-\mu/\gamma} \end{cases}$$

If any one of the three inequalities is reversed, it indicates the system is unstable. This is consistent with Theorem 2.1.1. Note that if peers depart immediately after obtaining a complete collection (i.e. $\gamma=\infty$), then the stability condition becomes

$$\begin{cases} \lambda_1+\lambda_2 < 2\lambda_3 \\ \lambda_2+\lambda_3 < 2\lambda_1 \\ \lambda_1+\lambda_3 < 2\lambda_2 \end{cases}$$

If $\lambda_1,\lambda_2,\lambda_3$ are not all equal, at least one equality is reversed, so the system is unstable. This special case when $\gamma=\infty$ is considered in [3].

2.3 Proof outline - missing piece syndrome

The analysis of the above three examples suggests that when we consider the system to be in heavy load, the worst distribution of load is that nearly all

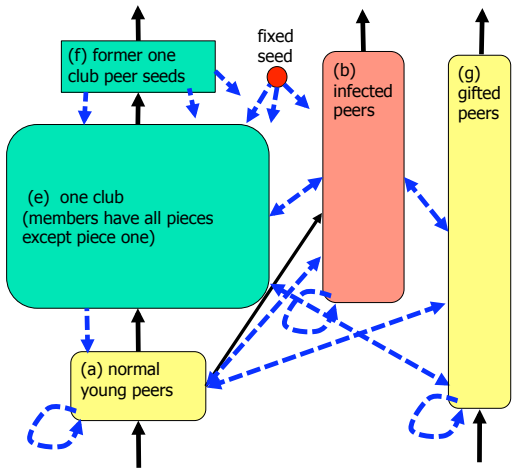


Figure 2.2: Solid lines show flow of peers and dashed lines show flow of pieces.

peers have the same type C with $|C| = K-1$. If the system is able to recover from that kind of heavy load, it can recover from other kinds of heavy load. With this intuition in mind, a sketch of proof of Theorem 2.1.1 is offered as follows.

First, we sketch the proof of Theorem 2.1.1(a) about transience when $0 < \mu < \gamma < \infty$. Without loss of generality, assume that (2.2) is true for $k = 1$, or equivalently, $\triangle_{\mathcal{F}\setminus\{1\}} > 0$. Consider the following partition of peers into five groups, as shown in Figure 2.2.

- *Normal young peer*: A normal young peer is a peer that does not have piece one and does not have at least one other piece.
- *Infected peer*: An infected peer is a peer that obtained piece one after arriving, but before obtaining all the other pieces. Once a peer is infected, it remains infected until it leaves the system; it is considered to be infected even when it is a peer seed.
- *Gifted peer*: A gifted peer is a peer that arrived with piece one. A gifted peer is gifted for its entire time in the system; it is considered to be gifted even when it is a peer seed.
- *One-club peer*: A one-club peer is a peer that has all pieces except piece one. That is, the one-club is the group of peers of type $\{2, 3, ...K\}$.

- *Former one-club peer*: A former one-club peer is a peer in the system that is not a one-club peer but at some earlier time was a one-club peer. Note that a former one-club peer is a peer seed. Infected peers and gifted peers can also become peer seeds.

Consider the system starting from an initial state in which there are many peers in the system, and all of them are one-club peers. The system evolves as shown in Figure 2.2. Piece one can arrive into the system from outside the system in two ways: uploads by the fixed seed or arrivals of gifted peers. Ignore momentarily the effect of normal young peers getting piece one (and becoming infected). Most of the uploads by the fixed seed are uploads of piece one to one-club peers. One such upload creates a new peer seed, which on average will upload piece one to about μ/γ more one-club peers, and each of those will upload piece one to about μ/γ more one-club peers, and so forth, in a branching process. Each upload of a piece by the fixed seed thus ultimately causes, on average, about $\frac{1}{1-\mu/\gamma}$ departures from the one-club. Each gifted peer, with type C on arrival, for some C with $1 \in C$, will directly upload to, on average, about $K - |C| + \mu/\gamma$ one-club peers, and those will become peer seeds who also could upload to about μ/γ more one-club peers, and so forth, so that the total expected number of one-club departures caused by the type C gifted peer is $(K - |C| + \mu/\gamma)\frac{1}{1-\mu/\gamma}$. Summing these quantities and subtracting them from the arrival rate of peers without piece one gives $\triangle_{\mathcal{F}\setminus\{1\}}$. So $\triangle_{\mathcal{F}\setminus\{1\}} > 0$ indicates that the arrival rates of peers missing piece one is larger than the upload rate of piece one, causing the one-club size to grow linearly with time.

The above analysis neglects the possibility that normal young peers can also receive piece one, creating infected peers. An infected peer can upload to one-club peers, creating former one-club peers, and to normal young peers, creating more infected peers. This results in a branching process comprised of infected peers and former one-club peers. By the theory of branching process, the expected number of infected offspring of a former one-club peer or an infected peer will converge to zero, as the fraction of one-club peers converges to one. Hence, when the one-club is large enough, the existence of infected peers does not appreciably affect the growth of the one-club. The detailed proof of transience is offered in Section 2.4.

Second, we sketch the proof of Theorem 2.1.1(b) about positive recurrence

for the case $0 < \mu < \gamma < \infty$ under the assumption that (2.4) is valid for all $S \in \mathcal{C} \setminus \{\mathcal{F}\}$. The above discussion suggests that when $\triangle_{\mathcal{F}\setminus\{1\}} < 0$, the departure rate of the one-club is larger than the arrival rate of peers missing piece one; therefore, the system has the ability to recover from a single heavy load in the one-club. Moreover, when $k = 2, 3, ...K$ and there is a single heavy load in the type $\mathcal{F} \setminus \{k\}$ group, similar reasoning suggests that the system can recover if $\triangle_{\mathcal{F}\setminus\{k\}} < 0$. To get better idea of the proof, here we consider other distributions of heavy load.

- Suppose there is a single heavy load in some type S group with $|S| \leq K - 2$. Uploads from the fixed seed (with rate U) and from new peers holding pieces not in S (with rate $\sum_{C:C\not\subseteq S} \lambda_C$) keep creating departures from the type S group. If we ignore the period of time from when a peer departs from the type S group until the same peer becomes a peer seed, we see that the average remaining lifetime of every peer who departs from the type S group is greater than or equal to $\frac{1}{\gamma}$. In this lifetime the peer uploads on average approximately μ/γ pieces to type S peers, which creates more departures from the type S group. Including the root, every departure from the type S group can ultimately cause at least $\frac{1}{1-\mu/\gamma}$ departures from the type S group, on average. Because every new type C peer with $C \not\subseteq S$ eventually uploads on average $K - |C| + \mu/\gamma$ pieces to type S peers, the departure rate of type S group is larger than $\left[U + \sum_{C:C\not\subseteq S} \lambda_C(K - |C| + \mu/\gamma)\right] \frac{1}{1-\mu/\gamma}$. Because peers mainly download pieces from type S peers, almost all new type C peers with $C \subseteq S$ ultimately join the type S group. So the near term arrival rate of type S group is less than but close to $\sum_{C:C\subseteq S} \lambda_C$, which is smaller than the aggregate departure rate of type S peers by (2.4). So the system can recover from the heavy load.

- Suppose there is a single heavy load in the type $\mathcal{F}$ group, that is, the group of peer seeds. The departure rate of peer seeds, $\gamma n_{\mathcal{F}}$, scales linearly with the number of peer seeds, $n_{\mathcal{F}}$, as in an infinite server queueing system. So the system can recover however large the group of peer seeds is.

- Suppose there are heavy loads in at least two groups of different types, say types C_1 and C_2. In this condition, either $C_1 \not\subseteq C_2$ or $C_2 \not\subseteq C_1$ is

true, so peers in at least one of the groups, say C_1, can upload pieces to peers in the other group, say C_2. The rate of peers departing from the type C_2 group is quite high, due to the large rate of uploads from type C_1 peers, so the system can quickly escape from that region of the state space.

The above paragraphs summarize how the system can recover from all distributions of heavy load. To provide a proof of stability it must also be shown that the load cannot spiral up without bound through some oscillatory behavior. For that we use a Lyapunov function and apply the Foster-Lyapunov stability criterion [57]. The detailed proof is offered in Section 2.5.

2.4 Proof of transience

In the following the detailed proof of Theorem 2.1.1(a) is given. It is obvious the system is transient if no copies of piece k can enter the system. Without loss of generality, assume $0 < \mu < \gamma \leq \infty$ and assume $\triangle_{\mathcal{F}\setminus\{1\}} > 0$. For a given time $t \geq 0$, define the following random variables, using the terminology of Section 2.3 and Figure 2.2:

- Y_t^a : number of normal young peers (group (a)) at time t.
- Y_t^b : number of infected peers (group (b)) at time t.
- Y_t^g : number of gifted peers (group (g)) at time t.
- Y_t^e : number of one-club peers (group(e)) at time t.
- Y_t^f : number of former one-club peers (group (f)) at time t.
- A_t : cumulative number of arrivals, up to time t, of peers without piece one at time of arrival.
- D_t : cumulative number of downloads of piece one, up to time t. (Peers arriving with piece one are not counted.)
- N_t : number of peers at time t.

The system is modeled by an irreducible, countable-state Markov chain. A property of such random processes is that either all states are transient,

or no state is transient. Therefore, to prove Theorem 2.1.1(a), it is sufficient to prove that some particular state is transient. With that in mind, we assume that the initial state is the one with N_o peers, and all of them are one-club peers, where N_o is a large constant specified below. Given a small number ξ with $0 < \xi < 1$, let τ be the extended stopping time defined by $\tau = \min\{t \geq 0 : Y_t^e + Y_t^f \leq (1-\xi)N_t\}$, with the usual convention that $\tau = \infty$ if $Y_t^e + Y_t^f > (1-\xi)N_t$ for all t. It suffices to prove that

$$P\{\tau = \infty \text{ and } \lim_{t\to\infty} N_t = +\infty\} > 0. \tag{2.5}$$

The probability of the event in (2.5) depends on only the out-going transition rates for states such that $Y^e + Y^f > (1-\xi)N$. Thus, we can and do prove (2.5) instead for an alternative system, that has the same initial state, and the same out-going transition rates for all states such that $Y^e + Y^f > (1-\xi)N$, as the original system. The alternative system, however, guarantees a lower bound on the rate of downloads by the set of peers in groups (e) and (f), and an upper bound on the aggregate rate of downloads of piece one by peers in group (a). This can be done so that the alternative system has the following properties:

1. A peer in group (a), (b), or (g) that is not yet a seed peer receives usable download opportunities at a rate greater than or equal to $(1-\xi)\mu$. (If $Y^e + Y^f > (1-\xi)N$, these opportunities can be provided by the peers in groups (e) and (f).)

2. A peer with a complete collection departs according to an exponentially distributed random variable with parameter γ.

3. Each peer in group (b), (g), or (f) uploads to the set of peers in (a) with rate at most $\mu\xi$.

4. The fixed seed uploads to the set of peers in group (a) with rate at most ξU.

5. Each peer in group (b), (g), or (f) uploads to the set of peers in group (e) with rate at most μ.

6. The fixed seed uploads to the set of peers in group (e) with rate at most U.

For the remainder of this proof we consider the alternative system, but for brevity of notation, we use the same notation for it as for the original system and refer to it as the original system.

Only peers in groups (a) and (e) download piece one; peers in the other three groups already have piece one. A peer in group (a) downloading piece one immediately moves to group (b), and a peer in group (e) downloading piece one immediately moves to group (f). Thus, a download of piece one creates either a group (b) peer or a group (f) peer. A group (b) peer or group (f) peer stays in the same group until it leaves the system. While a peer in group (b) or (f) is in the system it can generate more peers in groups (b) and (f) by uploading piece one, and those peers are considered to be offspring spawned by the peer. Since offspring can themselves spawn offspring, there is a branching process, and a group (b) or group (f) peer has a set of descendants.

We shall consider the evolution of a portion of the system under some statistical assumptions that are different from those in the original system. We refer to it as the *autonomous branching system* (ABS) because strong independence assumptions are imposed. The ABS pertains only to those peers that have piece one. It is shown below that the original system can be stochastically coupled to the ABS so that uploads of piece one happen in the original system only when they also happen in the ABS. We begin by considering only group (b) and group (f) peers. In the original system, a group (b) peer was formerly a group (a) peer, and a group (f) peer was formerly a group (e) peer; such previous history is irrelevant for the system under the ABS; the description below concerns such a peer only from the time it becomes a group (b) or group (f) peer. The statistical assumptions for the ABS involving these peers are as follows:

- A group (b) peer is required to download $K-1$ pieces; usable opportunities for such downloads arrive according to a Poisson process of rate $\mu(1-\xi)$. (The interpretation is that, when a group (b) peer appears, any piece it might have had besides piece one is ignored or discarded.) After the $K-1$ downloads, the group (b) peer remains in the system as a seed peer for a *seed dwell duration* that is exponentially distributed with parameter γ.

- A group (f) peer remains in the system for a seed dwell duration that

is exponentially distributed with parameter γ.

- A group (b) peer or group (f) peer spawns group (b) peers according to a Poisson process of rate $\xi\mu$ and it spawns group (f) peers according to a Poisson process of rate μ.

- The Poisson processes for spawning offspring, as well as the seed dwell durations, are mutually independent.

The above assumptions uniquely determine the distribution of the number of offspring, and therefore the total number of descendants, of a group (b) or group (f) peer. On average, a group (b) peer is in the system (as a group (b) peer) for $\frac{K-1}{(1-\xi)\mu} + \frac{1}{\gamma}$ time units, and thus on average a group (b) peer spawns $\xi(\frac{K-1}{1-\xi} + \frac{\mu}{\gamma})$ offspring of type (b) and $\frac{K-1}{1-\xi} + \frac{\mu}{\gamma}$ offspring in group (f). Similarly, on average, a peer in group (f) spawns $\frac{\xi\mu}{\gamma}$ offspring of type (b) and $\frac{\mu}{\gamma}$ offspring in group (f). Let m_b denote one plus the mean number of descendants of a group (b) peer and let m_f denote one plus the mean number of descendants of a group (f) peer, in the ABS. Then by the theory of branching processes, $\binom{m_b}{m_f}$ is the minimum nonnegative solution to the equations

$$\begin{pmatrix} m_b \\ m_f \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} \xi\left(\frac{K-1}{1-\xi} + \frac{\mu}{\gamma}\right) & \frac{K-1}{1-\xi} + \frac{\mu}{\gamma} \\ \frac{\xi\mu}{\gamma} & \frac{\mu}{\gamma} \end{pmatrix} \begin{pmatrix} m_b \\ m_f \end{pmatrix}.$$

The two-by-two matrix involved here has rank one, and the solution is easily found to be finite if

$$\xi\left(\frac{K-1}{1-\xi} + \frac{\mu}{\gamma}\right) + \frac{\mu}{\gamma} < 1. \tag{2.6}$$

If (2.6) holds,

$$\begin{pmatrix} m_b \\ m_f \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \frac{1+\xi}{1 - \xi(\frac{K-1}{1-\xi} + \frac{\mu}{\gamma}) - \frac{\mu}{\gamma}} \begin{pmatrix} \frac{K-1}{1-\xi} + \frac{\mu}{\gamma} \\ \frac{\mu}{\gamma} \end{pmatrix}$$

and, in addition, the second moment of the number of descendants of a peer of either group (b) or (f) is finite and monotonically increasing in ξ. Note that

$$\begin{pmatrix} m_b \\ m_f \end{pmatrix} \overset{\xi \to 0}{\to} \begin{pmatrix} \frac{K}{1-\frac{\mu}{\gamma}} \\ \frac{1}{1-\frac{\mu}{\gamma}} \end{pmatrix}.$$

Next, we extend the scope of the ABS to include a gifted peer; this entails the following statistical assumptions:

- A gifted peer with piece collection C upon arrival is required to download $K - |C|$ pieces; usable opportunities for such downloads arrive according to a Poisson process of rate $\mu(1-\xi)$. After the $K-1$ downloads, the group (b) peer remains in the system as a seed peer for a seed dwell duration that is exponentially distributed with parameter γ.

- While a gifted peer is in the system, it spawns group (b) peers according to a Poisson process of rate $\xi\mu$ and it spawns group (f) peers according to a Poisson process of rate μ.

- The Poisson processes for spawning offspring, as well as the seed dwell duration, are mutually independent.

The mean time a gifted peer with initial piece collection C is in the system is thus $\frac{K-|C|}{\mu(1-\xi)} + \frac{1}{\gamma}$, so the mean total number of descendants of a gifted peer (*not* including the gifted peer itself) is given by

$$m_g(C) = \left(\frac{K-|C|}{\mu(1-\xi)} + \frac{1}{\gamma}\right)(\xi\mu m_b + \mu m_f) = \left(\frac{K-|C|}{1-\xi} + \frac{\mu}{\gamma}\right)(\xi m_b + m_f)$$

Note that $m_g(C) \overset{\xi\to 0}{\to} \left(K - |C| + \frac{\mu}{\gamma}\right)\frac{1}{1-\frac{\mu}{\gamma}}$.

Finally, we extend the scope of the ABS to include the processes of arrivals of gifted peers and the uploads of the fixed seed; this entails the following assumptions:

- For each C with $1 \in C$, gifted peers with initial piece collection C arrive according to a Poisson process of rate λ_C (as in the original model).

- The fixed seed spawns peers in group (b) according to a Poisson process of rate ξU and it spawns peers in group (f) according to a Poisson process of rate U.

- The Poisson processes of arrivals are mutually independent.

- Offspring of the fixed seed and gifted peers are considered to be *root peers*. The evolution of the descendants of root peers are mutually independent.

Let $\widehat{D}_t$ denote the cumulative number of group (b) and group (f) peers appearing in the ABS up to time t.

Lemma 2.4.1 *The process $(D_t : t \geq 0)$ is stochastically dominated (see Definition A.3.1) by $(\widehat{D}_t : t \geq 0)$.*

Proof of Lemma 2.4.1 is provided in Appendix B.1. In the following, if proofs are not provided immediately following the results, they can be found in the appendix. Lemma 2.4.1 implies Corollary 2.4.1, which bounds D_t.

Corollary 2.4.1 *Given $\epsilon > 0$, if ξ is sufficiently small, then for all B sufficiently large,*

$$P\left\{D_t < B + \frac{\rho}{1-\mu/\gamma}t + \epsilon t \quad \textit{for all } t \geq 0\right\} \geq 0.9, \tag{2.7}$$

where $\rho = U + \sum_{C:1\in C} \lambda_C \left(K - |C| + \frac{\mu}{\gamma}\right)$.

The following lemma bounds A_t.

Lemma 2.4.2 *Given $\epsilon > 0$, if B is sufficiently large,*

$$P\left\{A_t > -B + \left(\sum_{C:k\notin C} \lambda_C - \epsilon\right) t \quad \textit{for all } t \geq 0\right\} \geq 0.9 \tag{2.8}$$

Proof. The process A is a Poisson process with rate $\sum_{C:k\notin C} \lambda_C$. Thus, (3.8) follows from Kingman's moment bound (see Proposition A.4.1 in the appendix.) □

Lemma 2.4.3 and corollary 2.4.2 bounds $Y_t^a + Y_t^b + Y_t^g$.

Lemma 2.4.3 *The process $Y_t^a + Y_t^b + Y_t^g$ is stochastically dominated by the number of customers in an $M/GI/\infty$ queueing system with initial state zero, arrival rate λ_{total}, and service times having mean $\frac{K}{\mu(1-\xi)} + \frac{1}{\gamma}$.*

Corollary 2.4.2 *Given $\epsilon_o > 0$ and $\xi > 0$, if B is sufficiently large,*

$$P\{Y_t^a + Y_t^b + Y_t^g < B + \epsilon_o t \quad \textit{for all } t \geq 0\} \geq 0.9 \tag{2.9}$$

Proof. The Corollary follows from Lemmas 2.4.3 and A.6.1 with m in Lemma A.6.1 equal to $\frac{K}{\mu(1-\xi)} + \frac{1}{\gamma}$ and ϵ equal to ϵ_o. □

The proof of Theorem 2.1.1(a) is now completed.

- Select $\epsilon > 0$ so that $2\epsilon < \triangle_{\mathcal{F}\setminus\{1\}}$.
- Select $\xi > 0$ so small that (3.11) holds for sufficiently large B.
- Select ϵ_o small enough that $\frac{\epsilon_o}{\triangle_{\mathcal{F}\setminus\{1\}}-2\epsilon} < \xi$.
- Select B large enough that (3.11), (3.8), and (3.10) hold.
- Select N_o large enough that $\frac{B}{N_o-2B} \leq \xi$.

Let $\mathcal{E}$ be the intersection of the three events on the left sides of (3.11), (3.8), and (3.10). By the choices of the constants, (3.11), (3.8), and (3.10) hold, so that $P\{\mathcal{E}\} \geq 0.7$. To complete the proof, it will be shown that $\mathcal{E}$ is a subset of the event in (2.5), thereby establishing (2.5). Since N_t is greater than or equal to the number of peers in the system that do not have piece one, on $\mathcal{E}$, $N_t \geq N_o + A_t - D_t > N_o - 2B + (\triangle_{\mathcal{F}\setminus\{1\}} - 2\epsilon)t$ for all $t \geq 0$. Therefore, on $\mathcal{E}$, for any $t \geq 0$,

$$\begin{aligned}\frac{Y_t^a + Y_t^b + Y_t^g}{N_t} &< \frac{B + \epsilon_o t}{N_o - 2B + (\triangle_{\mathcal{F}\setminus\{1\}} - 2\epsilon)t} \\ &\leq \max\left\{\frac{B}{N_o - 2B}, \frac{\epsilon_o}{\triangle_{\mathcal{F}\setminus\{1\}} - 2\epsilon}\right\} \leq \xi.\end{aligned}$$

Thus, $\mathcal{E}$ is a subset of the event in (2.5) as claimed. This completes the proof of Theorem 2.1.1(a).

2.5 Proof of positive recurrence

Theorem 2.1.1(b) is proved in this section. The first subsection treats the case $0 < \mu < \gamma \leq \infty$ and the second subsection treats the case $0 < \gamma \leq \mu$.

2.5.1 Proof of positive recurrence when $0 < \mu < \gamma \leq \infty$

The detailed proof of Theorem 2.1.1(b) when $0 < \mu < \gamma \leq \infty$ is given in this part. Assume $0 < \mu < \gamma \leq \infty$ and assume (2.4) is valid for all $S \in \mathcal{C} \setminus \{\mathcal{F}\}$.

For any nonnegative function $F = F(\mathbf{n})$ on the state space of the system, the drift of F at state $\mathbf{n}$ is defined as

$$Q(F)(\mathbf{n}) := \sum_{\mathbf{n}':\mathbf{n}'\neq\mathbf{n}} q(\mathbf{n},\mathbf{n}')\,[F(\mathbf{n}') - F(\mathbf{n})] \tag{2.10}$$

If, as usual, the diagonal elements $q(\mathbf{n},\mathbf{n})$ of the transition matrix Q are chosen so that row sums are zero, $Q(F)$ is the product of the matrix Q and function F, viewed as a vector. In this thesis, we apply the following lemma implied by the Foster-Lyapunov criterion.

Lemma 2.5.1 *The P2P Markov process is positive recurrent and $E[n] < +\infty$ in equilibrium, if there is a nonnegative function $W(\mathbf{n})$ on the state space of the process, such that (i) $\{\mathbf{n} : W(\mathbf{n}) \leq c\}$ is a finite set for any constant c, and (ii) there exists $n_0 \geq 0$ and $\xi > 0$ so that $QW \leq -\xi n < 0$ whenever $n \geq n_0$. We call such a W a valid Lyapunov function.*

Proof. For any $\mathbf{n}$, $q(\mathbf{n},\mathbf{n}')$ is nonzero for only finitely many values of $\mathbf{n}'$, so QW is finite for all $\mathbf{n}$. Therefore $\max_{\mathbf{n}:n<n_0} QW(\mathbf{n})$ is finite. The lemma follows from the combined Foster-Lyapunov stability criterion and moment bound — see Proposition A.1.1 in the appendix. □

The particular Lyapunov function we use in case $0 < \mu < \gamma < \infty$ is:

$$W \;:=\; \sum_{C:C\in\mathcal{C}} r^{|C|}T_C, \text{ with } T_C := \begin{cases} \frac{1}{2}E_C^2 + \alpha E_C\phi(H_C), & C \neq \mathcal{F} \\ \frac{1}{2}n^2, & C = \mathcal{F} \end{cases} \tag{2.11}$$

and in case $0 < \mu < \gamma = \infty$ is

$$W \;:=\; \sum_{C:C\in\mathcal{C}\setminus\{\mathcal{F}\}} r^{|C|}T_C, \tag{2.12}$$

with the following notation:

- $r \in (0,\frac{1}{2}), d \in (1,\infty), \beta \in (0,\frac{1}{2}), \alpha \in (\frac{1}{2},1)$ are positive constants to be specified, with r and β small, d large, and α close to one.
- $\mathcal{E}_C := \{C' : C' \subseteq C\}$, which is the collection of types of peers which are or can become type C peers.

- $\mathcal{H}_C := \{C' : C' \in \mathcal{C}, C' \not\subseteq C\}$, which is the collection of types of peers which can help type C peers. Notice that $\mathcal{F} \in \mathcal{H}_C$ for any $C \in \mathcal{C} \setminus \{\mathcal{F}\}$ and $\mathcal{H}_{\mathcal{F}} = \emptyset$.

- $E_C := \sum_{C':C' \in \mathcal{E}_C} n_{C'}$, $H_C := \frac{1}{1-\mu/\gamma} \sum_{C':C' \in \mathcal{H}_C} (K - |C'| + \mu/\gamma) n_{C'}$. e.g. $E_{\mathcal{F}} = n$ and $H_{\mathcal{F}} = 0$.

- ϕ is the function with parameters d, β, defined as

$$\phi(x) := \begin{cases} (2d + \frac{1}{2\beta} - x) & \text{if } 0 \le x \le 2d \\ \frac{\beta}{2}(x - 2d - \frac{1}{\beta})^2 & \text{if } 2d < x \le 2d + \frac{1}{\beta} \\ 0 & \text{if } x > 2d + \frac{1}{\beta} \end{cases} \tag{2.13}$$

  Thus $\phi'(x) = -1$ for $0 \le x \le 2d$, $\phi'(x) = 0$ for $x \ge 2d + 1/\beta$, and ϕ' increases linearly from -1 to 0 over the interval $[2d, 2d + 1/\beta]$. In particular, $-1 \le \phi'(x) \le 0$ for $x \ge 0$.

In the proof, we consider the following two classes of states, where ϵ is to be selected with $0 < \epsilon < \frac{1}{2}$. The classes overlap and their union includes every nonzero state:

Definition 2.5.1 *Class I is the set of states* $\mathbf{n}$ *such that there exists* $S \in \mathcal{C} \setminus \{\mathcal{F}\}$*, so that* $n_S/n > 1 - \epsilon$*; class II is the set of states* $\mathbf{n}$ *such that there exist* $C_1, C_2 \in \mathcal{C}$*, either* C_1 *and* C_2 *being distinct or both equal to* $\mathcal{F}$*, so that,* $n_{C_1}/n > \epsilon/2^K$ *and* $n_{C_2}/n > \epsilon/2^K$*.*

The main idea of the proof is to show that W is a valid Lyapunov function for an appropriate choice of $(r, d, \beta, \alpha, \epsilon)$. The given parameters of the network, $K, U, \lambda = (\lambda_S : S \in \mathcal{C}), \gamma$ and μ, are treated as constants. Functions on the state space are considered which may depend on the variables r, d, β, α and ϵ. It is convenient to adopt the big theta notation $\Theta(*)$, with the understanding that it is uniform in these variables; this is summarized in the following definitions.

Definition 2.5.2 *Given functions* f *and* g *on the state space, we say* $f = \Theta(g)$ *if there exist constants* $k_1, k_2, n_0 > 0$*, not depending on* $(r, d, \beta, \alpha, \epsilon)$*, such that* $k_1|g(\mathbf{n})| \le |f(\mathbf{n})| \le k_2|g(\mathbf{n})|$ *for all* $\mathbf{n}$ *such that* $n > n_0$*.*

For example, $2 \in \Theta(1), \lambda_{total} n \in \Theta(n)$, $d \notin \Theta(1)$, $d \in \Theta(d)$. Similarly, we adopt notions of "small enough" and "large enough" that are uniform in $(r, d, \beta, \alpha, \epsilon)$:

Definition 2.5.3 *The statement, "condition A is true if $x > 0$ is* small enough*", means there exists a constant $k > 0$, not depending on $(r, d, \beta, \alpha, \epsilon)$, such that A is true for any $x \in (0, k)$. Similarly, the statement, "condition A is true if $x > 0$ is* large enough*", means there exists a constant $k > 0$, not depending on $(r, d, \beta, \alpha, \epsilon)$, such that A is true for any $x \in (k, \infty)$.*

Some additional notation is applied for proofs in this chapter only:

- $M_\phi := 3d + \frac{1}{\beta}$. We have $M_\phi > \max_x \phi(x)$ and $M_\phi > \min\{x : \phi(x) = 0\} + d > 1$.
- For any $\mathcal{X}, \mathcal{X}' \subseteq \mathcal{C}$, $\Gamma_{\mathcal{X},\mathcal{X}'} := \sum_{C \in \mathcal{X}} \sum_{C' : C' \in \mathcal{X}'} \Gamma_{C,C'}$, where $\Gamma_{C,C'}$ is defined in (2.1).
- D_C is defined by

$$D_C := \begin{cases} \sum_{i : i \in \mathcal{F}} \Gamma_{C, C \cup \{i\}} & \text{if } C \neq \mathcal{F}, \\ \gamma n_{\mathcal{F}} & \text{if } C = \mathcal{F}, \gamma < \infty, \\ 0 & \text{if } C = \mathcal{F}, \gamma = \infty. \end{cases}$$

  Except in the case $C = \mathcal{F}$ and $\gamma = \infty$, D_C is the aggregate rate that peers leave the group of type C peers.

- For any $\mathcal{X} \subseteq \mathcal{C}$, $n_{\mathcal{X}} := \sum_{C : C \in \mathcal{X}} n_C$, $D_{\mathcal{X}} := \sum_{C : C \in \mathcal{X}} D_C$, $D_{total} := D_{\mathcal{C}}$, $\lambda_{\mathcal{X}} := \sum_{C : C \in \mathcal{X}} \lambda_C$, $\lambda^*_{\mathcal{X}} = \sum_{C : C \in \mathcal{X}} \lambda_C (K - |C| + \mu/\gamma)$.

Now we start to prove that W given by (2.23) or (2.12) is a valid Lyapunov function. The following proof applies if either $0 < \mu < \gamma < \infty$ or $0 < \mu < \gamma = \infty$, with differences being stated when necessary.

To begin, we identify a simple approximation to the drift of W. Notice that $Q(*)$ is linear, so if $0 < \mu < \gamma < \infty$,

$$Q(W) = \sum_{C : C \in \mathcal{C}} r^{|C|} Q(T_C), \text{ with } Q(T_C) = \begin{cases} \frac{1}{2} Q(E_C^2) + \alpha Q(E_C \phi(H_C)) & \text{if } C \neq \mathcal{F} \\ \frac{1}{2} Q(n^2) & \text{if } C = \mathcal{F} \end{cases}$$

If $0 < \mu < \gamma = \infty$,

$$Q(W) = \sum_{C : C \in \mathcal{C} \setminus \{\mathcal{F}\}} r^{|C|} Q(T_C)$$

Define ŁW, an approximation of $Q(W)$, as follows: If $0 < \mu < \gamma < \infty$,

$$\text{ŁW} := \sum_{C:C\in\mathcal{C}} r^{|C|}\text{ŁT}_C, \tag{2.14}$$
$$\text{with ŁT}_C := \begin{cases} E_C Q(E_C) + \alpha E_C Q(\phi(H_C)) & \text{if } C \neq \mathcal{F} \\ nQ(n) & \text{if } C = \mathcal{F} \end{cases}$$

If $0 < \mu < \gamma = \infty$,

$$\text{ŁW} : \ = \sum_{C:C\in\mathcal{C}\setminus\{\mathcal{F}\}} r^{|C|}\text{ŁT}_C \tag{2.15}$$

Lemma 2.5.2 $|Q(W) - \text{ŁW}| \leq M_\phi(D_{total} + 1)\Theta(1)$.

Lemma 2.5.2 provides a bound on the estimation error. Now we offer Lemma 2.5.3 and Lemma 2.5.4, both concerning upper bounds of ŁT_C. They are applied for the proof of Lemma 2.5.5.

Lemma 2.5.3 *If d is large enough, $Q(E_C) \leq \Theta(1), Q(\phi(H_C)) \leq M_\phi\Theta(1)$, $\text{ŁT}_C \leq M_\phi\Theta(E_C) \leq M_\phi\Theta(n)$ for any $C \in \mathcal{C}$.*

Lemma 2.5.4 *If d is large enough, $1 - \alpha, \epsilon M_\phi, \beta$ are small enough and $\beta\left(\frac{K+\mu/\gamma}{1-\mu/\gamma}\right)^2 \leq \frac{1}{\alpha} - 1$, for any $S \in \mathcal{C} \setminus \{\mathcal{F}\}$ and any nonzero state $\mathbf{n}$ such that $n_S/n > 1 - \epsilon$,*

$$\text{ŁT}_S \leq \frac{1}{2}\triangle_S E_S \tag{2.16}$$

Here we provide a remark to explain Lemma 2.5.4. The proof of Lemma 2.5.4 is provided in the appendix.

Remark: Recall that $\text{ŁT}_S = E_S[Q(E_S) + \alpha Q(\phi(H_S))]$, where the term $E_S Q(E_S)$ can be traced back to the quadratic term $\frac{1}{2}E_S^2$ of W, and $\alpha E_S Q(\phi(H_S))$ can be traced back to the term $\alpha E_S Q(\phi(H_S))$ of W. Before giving the proof of Lemma 2.5.4, we describe why the term $\alpha E_S Q(\phi(H_S))$ is needed and how it helps ŁT_S be negative. It has been discussed that the worst distribution of heavy load is when the heavy load aggregates in a type with only one missing piece. Consider the case $|S| = K - 1$. Notice that $E_S Q(E_S) = E_S(\lambda_{\mathcal{E}_S} - \Gamma_{\mathcal{E}_S,\mathcal{H}_S})$ and $\Gamma_{\mathcal{E}_S,\mathcal{H}_S} \geq D_S \geq \frac{n_S}{n}[U + H_S\mu\frac{1-\mu/\gamma}{K+\mu/\gamma}]$. Here we assume $\frac{n_S}{n} \geq 1 - \epsilon$. So $\Gamma_{\mathcal{E}_S,\mathcal{H}_S}$ increases almost proportionally to H_S. When H_S is

larger than d for d sufficiently large, $\Gamma_{\mathcal{E}_S,\mathcal{H}_S}$ is larger than $\lambda_{\mathcal{E}_S}$, so $E_S Q(E_S)$ is negative and is bounded above by $-\Theta(E_S) = -\Theta(n)$. But when H_S is smaller than d, $\Gamma_{\mathcal{E}_S,\mathcal{H}_S}$ can be smaller than $\lambda_{\mathcal{E}_S}$, so $E_S Q(E_S)$ is positive and is lower bounded by $\Theta(E_S) = \Theta(n)$, which has the wrong sign. The term $\alpha E_S Q(\phi(H_S))$ is chosen so that $\alpha Q(\phi(H_S))$ balances out the coefficient $\lambda_{\mathcal{E}_S} - \Gamma_{\mathcal{E}_S,\mathcal{H}_S}$ when H_S is small, so that $\L T_S$ is still negative and upper bounded by $-\Theta(E_S)$.

The definition of H_S implies that, when n_S is close to n, H_S is the mean number of type S peers that will be helped by the helping peers, which are the ones in $\mathcal{H}_S$. (By saying a peer is helped, we mean a piece is uploaded to the peer.) In other words, H_S is the stored potential for helping type S peers. As type S peers are helped by the helping peers, the potential decreases, with the magnitude of decrease equal to the number of type S peers which are helped. So if we only consider the piece transmissions involving one peer of type S and one peer of type in $\mathcal{H}_S$, the downward drift of H_S has magnitude less than or equal to the downward drift of E_S. If we only consider the external arrivals and the uploads from the fixed seed, the terms in the drift of H_S are $\frac{1}{1-\mu/\gamma}\left[\sum_{C:C\in\mathcal{H}_S}(K-|C|+\mu/\gamma)\lambda_C + U\mu/\gamma\right]$, and the terms in the drift of E_S are $\lambda_{\mathcal{E}_S} - U$, the former is larger than the latter precisely because of (2.4). Finally, H_S has a bit more downward drift due to peers other than type S peers uploading to peers in $\mathcal{H}_S$, but that is small for ϵ sufficiently small. Combining the downward and the other drifts, we see that the drift of H_S is approximately the same as the drift of E_S, with the drift of H_S a little greater. The difference of the two drifts is $\triangle_S$, defined in (2.4). Also, when H_S is small, the function ϕ at H_S has derivative -1. Thus the coefficient of E_S in $\L T_S$, which is $Q(E_S) + \alpha Q(\phi(H_S))$, is negative because α is close to 1, so $\L T_S$ is upper bounded by $-\Theta(E_S) = -\Theta(n)$.

In sum, the above explains the reason we included the term $E_S\phi(H_S)$ in the Lyapunov function; it balances out the positive drift of $\frac{1}{2}E_S^2$ when H_S is small. ■

Lemmas 2.5.3 and 2.5.4 will be applied to prove the following lemma.

Lemma 2.5.5 *If d is large enough, $(1-\alpha), \beta, rM_\phi, \epsilon M_\phi r^{-K}$ are small enough, and $\beta\left(\frac{K+\mu/\gamma}{1-\mu/\gamma}\right)^2 \le \frac{1}{\alpha} - 1$,*

(a) On class I, $\L W \le -r^K\Theta(n)$;

(b) On class II, $\text{Ł}W \leq -r^K \epsilon^3 \Theta(n^2) + M_\phi \Theta(n)$.

With Lemmas 2.5.2 and 2.5.5, we now complete the proof of Theorem 2.1.1(b) in the case $0 < \mu < \gamma \leq \infty$.

On class I,

$$\begin{aligned} D_{total} &\leq D_S + \sum_{C:C\neq S} D_C \leq U + n_{\mathcal{H}_S}\mu + \sum_{C:C\neq S} \frac{n_C}{n}(U + n\mu) \\ &\leq 2(U + \epsilon n\mu) \in \Theta(1) + \epsilon\Theta(n) \end{aligned}$$

So Lemma 2.5.2 implies that on class I,

$$|Q(W) - \text{Ł}W| \leq \epsilon M_\phi \Theta(n) + M_\phi \Theta(1)$$

Combing with Lemma 2.5.5(a), implies that under the conditions of Lemma 2.5.5, on class I,

$$\begin{aligned} Q(W) &\leq \text{Ł}W + |Q(W) - \text{Ł}W| \leq -r^K \Theta(n) + \epsilon M_\phi \Theta(n) + M_\phi \Theta(1) \\ &\in -r^K \Theta(n) + M_\phi \Theta(1). \end{aligned} \tag{2.17}$$

if $\epsilon M_\phi r^{-K}$ is small enough.

On class II, $D_{total} \leq U + n\mu \in \Theta(n)$, so Lemma 2.5.2 implies that

$$|Q(W) - \text{Ł}W| \leq M_\phi \Theta(n)$$

Combining with Lemma 2.5.5(b) implies that under the conditions of Lemma 2.5.5, on class II,

$$Q(W) \leq \text{Ł}W + |Q(W) - \text{Ł}W| \leq -r^K \epsilon^3 \Theta(n^2) + M_\phi \Theta(n) \tag{2.18}$$

Equations (2.17) and (3.7) imply that if $(r, d, \beta, \alpha, \epsilon)$ satisfies the conditions of Lemma 2.5.5, there exists $\xi > 0$ sufficiently small such that $Q(W) \leq -\xi n$ for all n larger than some constant. For such ξ and such (r, d, β, α), W is a valid Lyapunov function, so by Lemma 2.5.1, Theorem 2.1.1(b) for the case $0 < \mu < \gamma \leq \infty$ is proved.

2.5.2 Proof of positive recurrence when $0 < \gamma \leq \mu$

Now we consider the case when $0 < \gamma \leq \mu$. Assume $U + \sum_{C:k\in C} \lambda_C > 0$ for all $k \in \mathcal{F}$. Then $U + \lambda^*_{\mathcal{H}_C} > 0$ for any $C \in \mathcal{C} \setminus \{\mathcal{F}\}$. Consider a Lyapunov function of the following form:

$$W' := \sum_{C:C\in\mathcal{C}} r^{|C|} T'_C, \text{ with } T'_C := \begin{cases} \frac{1}{2}E_C^2 + pE_C\phi(H'_C) & \text{if } C \neq \mathcal{F} \\ \frac{1}{2}n^2 & \text{if } C = \mathcal{F} \end{cases} \tag{2.19}$$

where $H'_C := \sum_{C':C'\in\mathcal{H}_C}(K + 1 - |C'|)n_{C'}$ and p is a constant (i.e. $p \in \Theta(1)$) such that

$$\lambda_{\mathcal{E}_C} - p(U + \lambda^*_{\mathcal{H}_C}) < 0, \forall C \in \mathcal{C} \setminus \{\mathcal{F}\}. \tag{2.20}$$

The variable α is not used in this section, so the big Θ notation is uniform in $(r, \beta, d, , \epsilon)$.

Define $ŁW'$, as follows:

$$ŁW' := \sum_{C:C\in\mathcal{C}} r^{|C|} ŁT'_C, \tag{2.21}$$

$$\text{with } ŁT'_C := \begin{cases} E_C Q(E_C) + pE_C Q(\phi(H'_C)) & \text{if } C \neq \mathcal{F} \\ nQ(n) & \text{if } C = \mathcal{F} \end{cases}$$

Lemmas 2.5.2 and 2.5.3 can be verified as before, with H_C, W, $ŁW$, and $ŁT_C$ replaced by H'_C, W', $ŁW'$, and $ŁT'_C$, respectively. The following lemma similar to Lemma 2.5.4 can be established:

Lemma 2.5.6 *If d is large enough, $\epsilon M_\phi, \beta$ are small enough, for any $S \in \mathcal{C} - \{\mathcal{F}\}$ such that $n_S/n > 1 - \epsilon$,*

$$ŁT'_S \leq \frac{1}{2}[\lambda_{\mathcal{E}_S} - p(U + \lambda^*_{\mathcal{H}_S})]E_S. \tag{2.22}$$

With Lemma 2.5.2, 2.5.3 and 2.5.6, Lemma 2.5.5 with $ŁW$ replaced by $ŁW'$ can be easily verified to be valid. Thereby Theorem 2.1.1(b) at condition $0 < \gamma \leq \mu$ is proved.

2.6 General piece selection policies

A piece selection policy is used to choose which piece is transferred whenever one peer or the fixed seed is to upload a piece to a chosen peer. The random useful piece selection policy is assumed in Theorem 2.1.1, but the theorem can be extended to a large class of piece selection policies. Such extension was noted in [12] for the less general model of that paper. Essentially the only restriction needed is that if the uploading peer or fixed seed has a useful piece for the downloading peer, then a useful piece must be transferred. This restriction is similar to a work-conserving restriction in the theory of service systems. In particular, Theorem 2.1.1 extends to cover a broad class of rarest first piece selection policies. Peers can estimate which pieces are more rare in a distributed way, by exchanging information with the peers they contact. Even more general policies would allow the piece selection to depend in an arbitrary way on the piece collections of all peers.

To be specific, consider the following family $\mathcal{H}$ of piece selection policies. Each policy in $\mathcal{H}$ corresponds to a mapping h from $\mathcal{C} \times (\mathcal{C} \cup \{\mathcal{F}\}) \times \mathcal{S}$ to the set of probability distributions on $\mathcal{F}$, satisfying the usefulness constraint:

$$\sum_{i \in B \setminus A} h_i(A, B, \mathbf{x}) = 1 \quad \text{whenever } B \not\subset A,$$

with the following meaning of h:

- When a type A peer is to download a piece from a type B peer and the state of the entire network is $\mathbf{x}$, piece i is selected with probability $h_i(A, B, \mathbf{x})$, for $i \in \mathcal{F}$.
- When a type A peer is to download a piece from the fixed seed and the state of the entire network is $\mathbf{x}$, piece i is selected with probability $h_i(A, \mathcal{F}, \mathbf{x})$, for $i \in \mathcal{F}$.

Theorem 2.1.1 can be extended to piece selection policies in $\mathcal{H}$. One minor change is needed, because the Markov process may not be irreducible for some piece selection policies. In general, the set of all states that are reachable from the empty state is the unique minimal closed set of states, and the process restricted to that set of states is irreducible. For example, if the lowest numbered useful piece is selected at each download opportunity, then the minimal closed set of states consists of the states such that each peer has

either no pieces or a consecutively numbered set of pieces beginning with the first piece. We state the result as a theorem.

Theorem 2.6.1 *(Stability conditions for general useful piece selection policies) Consider the network model of Section 2.1, except with the random piece selection policy replaced by a policy h in $\mathcal{H}$. (a) If either of the two conditions in Theorem 2.1.1(a) hold, then the Markov process is transient, and the number of peers in the system converges to infinity with probability one. (ii) If either of the two conditions in Theorem 2.1.1(b) hold, then the Markov process restricted to the closed set of states is positive recurrent, the mean time to reach the empty state from any initial state has finite mean, and the equilibrium distribution π is such that $\sum_{\mathbf{x}} \pi(\mathbf{x})|\mathbf{x}| < \infty$.*

Thus, with the possible exception of the borderline case, rarest first piece selection does not increase the region of stability.

2.7 Stability region under network coding

Network coding, introduced by Ahlswede, Cai, and Yeung, [41], can be naturally incorporated into P2P distribution networks, as noted in [42]. The related work [58] considers all to all exchange of pieces among a fixed population of peers through random contacts and network coding. The method can be described as follows. The file to be transmitted is divided into K data pieces, $m_1, m_2, \ldots, m_K$. The data pieces are taken to be vectors of some fixed length r over a finite field $\mathbb{F}_q$ with q elements, where q is some power of a prime number. If the piece size is M bits, this can be done by viewing each message as an $r = \lceil M/\log_2(q) \rceil$ dimensional vector over $\mathbb{F}_q$. Any coded piece e is a linear combination of the original K data pieces: $e = \sum_{i=1}^{K} \theta_i m_i$; the vector of coefficients $(\theta_1, \ldots, \theta_K)$ is called the *coding vector* of the coded piece; the coding vector is included whenever a coded piece is sent. Suppose the fixed seed uploads coded pieces to peers, and peers exchange coded pieces. In this context, the type of a peer A is the subspace V_A of $\mathbb{F}_q^K$ spanned by the coding vectors of the coded pieces it has received. Once the dimension of V_A reaches K, peer A can recover the original message. Let $\mathcal{V}$ denote the set of all subspaces of $\mathbb{F}_q^K$, so $\mathcal{V}$ is the set of possible types.

When peer A contacts peer B, suppose peer B sends peer A a random

linear combination of its coded pieces, where the coefficients are independent and uniformly distributed over $\mathbb{F}_q$. Equivalently, the coding vector of the coded piece sent from B is uniformly distributed over V_B. The coded piece is considered useful to A if adding it to A's collection of coded pieces increases the dimension of V_A. Equivalently, the piece from B is useful to A if its coding vector is not in the subspace $V_A \cap V_B$. The probability the piece is useful to A is therefore given by

$$\begin{aligned} &P\{\text{piece from } B \text{ is useful to } A\} = 1 - \frac{|V_A \cap V_B|}{|V_B|} \\ &= 1 - q^{dim(V_A \cap V_B) - dim(V_B)}. \end{aligned}$$

If peer B can possibly help peer A, meaning $V_B \not\subset V_A$ (true, for example, if $dim(V_B) > dim(V_A)$), then the probability that a random coded piece from B is helpful to A is greater than or equal to $1 - \frac{1}{q}$. Similarly, the probability a random coded piece from the seed is useful to any peer A with $dim(V_A) \leq K - 1$ is also greater than or equal to $1 - \frac{1}{q}$.

The network state $\mathbf{x}$ specifies the number of peers in the network of each type. There are only finitely many types, so the overall state space is still countably infinite. Moreover, the Markov process is easily seen to be irreducible. A proof of the following variation of Theorem 2.1.1 is summarized below. Let $\widetilde{\mu} = \left(1 - \frac{1}{q}\right)\mu$.

Theorem 2.7.1 *(Stability conditions for a network coding based system) Suppose random linear network coding with vectors over $\mathbb{F}_q^K$ is used, with random peer contacts and parameters K, q, $(\lambda_V : V \in \mathcal{V})$, U, γ, and μ. Suppose $\lambda_{\mathbb{F}_q^K} = 0$ if $\gamma = \infty$, and $\lambda_{total} > 0$.*
(a) The Markov process is transient if either of the following two conditions is true:

- $0 < \mu < \gamma \leq \infty$ *and for some* $V^- \in \mathcal{V}$ *with* $dim(V^-) = K - 1$,

$$\lambda_{total} > \frac{U + \sum_{V \not\subset V^-} \lambda_V (K - dim(V) + 1)}{1 - \frac{\mu}{\gamma}};$$

- $0 < \gamma \leq \mu$, $U = 0$, *and* $\{V \in \mathcal{V} : \lambda_V > 0\}$ *does not span* $\mathbb{F}_q^K$.

(b) The process is positive recurrent and $E[n] < \infty$ in equilibrium, if either of the following two conditions is true:

- $0 < \widetilde{\mu} < \gamma \leq \infty$ *and for any* $V^- \in \mathcal{V}$ *with* $dim(V^-) = K - 1$,

$$\lambda_{total} < \left[U + \sum_{V: V \not\subset V^-} \lambda_V \left(K - dim(V) + \frac{q}{q-1} \right) \right] \times \left(\frac{1 - \frac{1}{q}}{1 - \frac{\widetilde{\mu}}{\gamma}} \right);$$

- $0 < \gamma \leq \widetilde{\mu}$ *and either* $U > 0$ *or* $\{V \in \mathcal{V} : \lambda_V > 0\}$ *spans* $\mathbb{F}_q^K$.

The gap between the necessary and sufficient conditions in Theorem 2.7.1 can be made arbitrarily small by taking q large enough.

For the case that peers arrive with pieces, network coding is quite effective at reducing the impact of the missing piece syndrome. For example, suppose peers with no pieces arrive at rate λ_0 and peers with one piece arrive at rate λ_1, where the coding vector for the piece given to a peer at time of arrival is uniformly distributed over all q^K possibilities. (So with probability q^{-K} the coding vector is the all zero vector and the piece is useless.) Suppose there are no other arrivals, that $U = 0$, and $\gamma = \infty$. Thus, the total arrival rate is $\lambda_{total} = \lambda_0 + \lambda_1$, and the fraction of peers arriving with one (possibly useless) piece is $f = \frac{\lambda_1}{\lambda_0 + \lambda_1}$. Then Theorem 2.7.1 yields that the Markov process is transient if $f < \frac{q}{(q-1)K}$ and positive recurrent if $f > \frac{q^2}{(q-1)^2 K}$. For example, if $q = 64$ and $K = 200$, the Markov process is transient if $f \leq \frac{1.014}{K} = 0.00507$ and positive recurrent if $f \geq \frac{1.032}{K} = 0.00516$. In contrast, without network coding and a fraction f of peers arriving with one uniformly randomly selected data piece, Theorem 2.1.1 implies the network is transient for any $f < 1$.

We comment briefly on how the proof of Theorem 2.1.1 can be modified to yield Theorem 2.7.1. First, consider how the proof of Theorem 2.1.1(a) can be modified to prove Theorem 2.7.1(a). Consider the main case, $0 < \mu < \gamma \leq \infty$. Let V^- be the subspace of $\mathbb{F}_q^K$ with dimension $K - 1$ appearing in part (a). To incorporate network coding, the partition of peers described in Section 2.3 should be replaced by the following partition:

- *Normal young peer*: A normal young peer is a peer A such that V_A is a proper subset of V^-.

- *Infected peer*: An infected peer is a peer B that was a normal young peer when it first arrived, but at the current time, $V_B \not\subset V^-$.

- *Gifted peer*: A gifted peer is a peer G such that at the time of its arrival, $V_G \not\subset V^-$.
- *One-club peer*: A peer of type V^-.
- *Former one-club peer*: A former one-club peer is a peer in the system that is not a one-club peer but at some earlier time was a one-club peer.

For any $N_o \geq 1$, it is possible to reach the state with N_o one-club peers and no other peers in the network.

Call a peer A *enlightened* if $V_A \not\subset V^-$. Note that gifted peers are enlightened when they arrive, and every other peer must become enlightened before departing. A peer becoming enlightened with network coding is analogous to a peer downloading the missing piece without network coding. In particular, for the proof of Theorem 2.7.1, the process D_t should be the cumulative number of downloads causing the recipient peers to become enlightened.

The same autonomous branching system (ABS) can be used as in the proof of Theorem 2.1.1. Lemma 3.6.4 remains true, but the coupling argument used to prove it becomes more subtle. The issue is that the rate that a group (b) or (g) peer downloads pieces can be less than $\mu(1-\xi)$, because random linear combinations are sent that are not always useful. This effect causes the group (b) and group (g) peers to remain in the system longer, so that they can continue to upload useful pieces to one-club peers for longer. However, note that if A is a group (b) or (g) peer that is not a peer seed, and B is a one-club peer, then the probability a random piece from A is useful to B is less than or equal to the probability a random piece from B is useful to A. Therefore, if the internal clocks of the group (b) and (g) peers are slowed down so that their download rate of useful pieces matches that of the original system, then their upload rate of useful pieces to the one-club peers will still be at least as large as in the original system.

The other parts of the proof of Theorem 2.1.1(a) readily carry over to imply Theorem 2.7.1(a).

Next, the modifications of Theorem 2.1.1(b) needed to prove Theorem 2.7.1(b) are described. The same approach works with the same form of Lyapunov function, except $\mathcal{V}$ is used as the set of types instead of $\mathcal{C}$. In places the cardinality $|C|$ of a type C is used in the proof of Theorem 2.1.1(b), the

dimension $dim(V)$ of a type V is used in the proof of Theorem 2.7.1(b). In some of the places that μ is used in the proof of Theorem 2.1.1(b), it should be replaced by $\widetilde{\mu}$.

The condition (2.23) holding for all $V^- \in \mathcal{V}$ is equivalent to the following: for any $S \in \mathcal{V} - \mathbb{F}_q^K$, $\triangle_S < 0$, where

$$\triangle_S = \sum_{V:V\subseteq S, V\in\mathcal{V}} \lambda_V - \left[U + \sum_{V:V\not\subseteq S, V\in\mathcal{V}} \lambda_V \left(K - dim(V) + \frac{\mu}{\gamma}\right)\right] \times \left(\frac{1-\frac{1}{q}}{1-\frac{\widetilde{\mu}}{\gamma}}\right).$$

The condition $\triangle_S < 0$ means that the rate of arrival of peers that can become type S peers is less than a lower bound on the long term rate that peers of type S receive useful pieces. The particular Lyapunov function we use in case $0 < \widetilde{\mu} < \gamma < \infty$ is:

$$W := \sum_{V:V\in\mathcal{V}} r^{dim(V)} T_V, \tag{2.23}$$

where

$$T_V := \begin{cases} \frac{1}{2}E_V^2 + \alpha E_V \phi(H_V) & \text{if } V \neq \mathbb{F}_q^K \\ \frac{1}{2}n^2 & \text{if } V = \mathbb{F}_q^K \end{cases}.$$

with α, r, d, β, and d and the function ϕ as in the proof of Theorem 2.1.1(b), and

- $\mathcal{E}_V := \{V' : V' \subseteq V\}$, which is the collection of types of peers which are or can become type V peers.
- $\mathcal{H}_V := \{V' : V' \in \mathcal{V}, V' \not\subseteq V\}$, which is the collection of types of peers which can help type V peers. Notice that $\mathbb{F}_q^K \in \mathcal{H}_V$ for any $V \in \mathcal{V} - \mathbb{F}_q^K$ and $\mathcal{H}_{\mathbb{F}_q^K} = \emptyset$.
- $E_V := \sum_{V':V'\in\mathcal{E}_V} x_{V'}$,
- $H_V := \left(\frac{1-\frac{1}{q}}{1-\widetilde{\mu}/\gamma}\right) \sum_{V':V'\in\mathcal{H}_V} (K - dim(V') + \mu/\gamma) x_{V'}$.

The proof that this Lyapunov function works for proving Theorem 2.7.1(b) parallels the proof of Theorem 2.1.1(b).

When network coding is considered, it is typically assumed that peers do not exchange descriptions of the pieces they already have. This is likely because such descriptions are more complex than simple bit vectors indicating

data pieces used without network coding, and because network coding works quite well even without such exchange. If exchange of information were used, then any time a peer A with subspace V_A transfers a piece to a peer B with subspace V_B such that $V_A \not\subset V_B$, a useful transfer could be achieved. Theorem 2.7.1 remains true under this mode of operation if $\widetilde{\mu} = \mu$ and $q \to \infty$ is taken in part (b), and the gap between parts (a) and (b) shrinks to zero.

2.8 The borderline of stability

Theorem 2.1.1 provides a sufficient condition for stability and a matching sufficient condition for instability, but it leaves open the borderline case namely, when equality holds in (2.3) (or, equivalently, (2.2)) for one or more values of $k \in \mathcal{F}$ and the strict inequality (2.3) holds for all other k. While it may not be interesting from a practical point of view, we comment on the borderline case. First, we give a precise result for a limiting case of the original system, and then we offer a conjecture. As in [12], a simpler network model results by taking a limit as $\mu \to \infty$. Call a state *slow* if all peers in the system have the same type, which includes the state such that there are no peers in the system. Otherwise, call a state *fast*. The total rate of transition out of any slow state does not depend on μ, and the total rate out of any fast state is bounded below by a positive constant times μ. For very large values of μ, the process spends most of its time in slow states. The original Markov process can be transformed into a new one by *watching* the original process while it is in the set of slow states. This means removing the portions of each sample path during which the process is in fast states, and time-shifting the remaining parts of the sample path to leave no gaps in time. The limiting Markov process, which we call the $\mu = \infty$ process, is the weak limit (defined as usual for probability measures on the space of càdlàg sample paths equipped with the Skorohod topology) of the original process watched in the set of slow states, as $\mu \to \infty$. If γ is fixed as $\mu \to \infty$ the model becomes degenerate, because a single peer seed would quickly convert all other peers into peer seeds. If $\gamma = \theta\mu$ for fixed θ and $\mu \to \infty$, the model under $\mu = \infty$ is more interesting but somewhat complicated. So we consider $\gamma = \infty$ for simplicity. For further simplicity we consider networks of the form in Example 3 (for any $K \geq 2$) for symmetric arrival rates. Thus $\lambda_C = \lambda$ if $|C| = 1$ and $\lambda_C = 0$

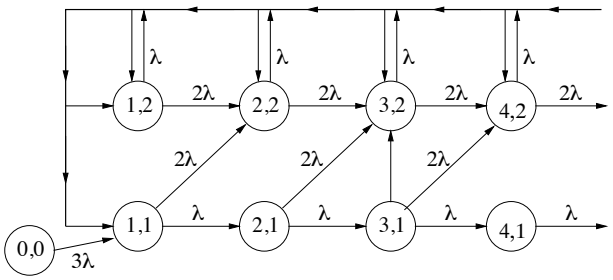


Figure 2.3: Transition rates of the $\mu = \infty$ variation of Example 3 with $\lambda_i = \lambda$ for all i.

otherwise. Also, $U = 0$ (no fixed seed) and $\gamma = \infty$. Note that these networks are borderline cases, not covered by Theorem 2.1.1.

By symmetry of the model, the state space of the $\mu = \infty$ process can be reduced to $\widehat{\mathcal{S}} = \{(0,0)\} \cup \{(n,k) : n \geq 1, 1 \leq k \leq K-1\}$, where a state (n,k) corresponds to n peers in the system which all possess the same set of k pieces. State (0,0) is transient. The transition rate diagram is pictured in Figure 2.3 for $K = 3$. States of the form $(n, K-1)$ form the top layer of states, and are those for which all peers have the same set of $K-1$ pieces. The transition out of such a state $(n, K-1)$ is described as follows. There is a transition to state $(n+1, K-1)$ with rate $(K-1)\lambda$, corresponding to the arrival of a new peer possessing one of the $K-1$ pieces that the other peers already have; the new peer instantly obtains all of the other $K-1$ pieces from the other peers. At rate λ a new peer arrives with the piece missing by all the other peers. The new peer downloads and uploads at equal rates, until it either obtains all the $K-1$ other pieces, or until all the other peers have departed. By the nature of Poisson processes, the probability distribution of the next state can be described in terms of flips of a fair coin, with "heads" denoting an upload by the new peer and "tails" denoting a download by the new peer. Let Z denote the number of "heads" in an experiment of repeated coin flips, when a fair coin is flipped until "tails" is observed $K-1$ times. Then Z represents the potential number of peers already in the system that can leave due to uploads from the new peer. If $Z \leq n-1$, then the next state is $(n-Z, K-1)$. If $Z \geq n$ then the new state will have the form $(1,j)$ with $1 \leq j \leq K-1$, corresponding to the case that all peers that were originally in the system depart, and the new peer remains. (The distribution over j can be computed easily but is not important.) Note that $E[Z] = K-1$, so the rate $(K-1)\lambda$ of upward unit jumps is equal to the mean rate $\lambda E[Z]$ of decrease due to

downward jumps (ignoring the lower boundary). Thus, when the process is in the top layer of states, it evolves as a stationary, independent increment process with zero drift. Such processes are null-recurrent, and therefore, *the* $\mu = \infty$ *process is null-recurrent.*

In essence, the $\mu = \infty$ process is simple because peers remain young for only an instant; there are no infections of young peers by gifted peers. If μ is finite, such infections effectively increase the departure rate, by roughly a constant divided by the number of peers in the system. The constant is decreasing in μ. A reflecting Brownian motion with negative drift inversely proportional to the state is positive recurrent if the constant of proportionality is sufficiently large, and is null recurrent otherwise. So the use of a diffusion approximation leads us to pose the following conjecture, which pertains to the symmetric flat-network model considered in [3]:

Conjecture 2.8.1 *Let* $K \geq 1$ *and suppose* $\lambda_C = \lambda$ *for* $|C| = 1$ *and* $\lambda_C = 0$ *otherwise. For some* $a_K > 0$, *the process is positive recurrent if* $0 < \mu/\lambda < a_K$ *and is null recurrent if* $\mu/\lambda > a_K$.

Chapter 3

MULTIPLE SWARMS FILE DISTRIBUTION

3.1 Model and the stability region

In this chapter, we establish the stability region of multiswarm networks, which model the case where multiple files are bundled together for P2P distribution.

Consider a BitTorrent-like file-sharing system, similar to the model in Section 2.1, but consisting of multiple files instead of one single file. Suppose all files are divided into *pieces* of equal size. We denote by $\mathcal{F} = \{1, 2, \ldots, K\}$ the set of all pieces of all files. A distinguished peer, the seed, is always present and holds $\mathcal{F}$. Represent a *file* C to be a non-empty subset of $\mathcal{F}$: $\emptyset \neq C \subseteq \mathcal{F}$. We refer to the set of peers interested in downloading file C as *Swarm* C. Each peer maintains a cache to store pieces it downloads. Peers arrive with empty caches, and each peer's cache is large enough to hold $\mathcal{F}$. Assume peers with empty caches arrive to Swarm C according to a Poisson process with rate $\lambda_{\emptyset,C}$, independent across swarms. We do *not* require different swarms to be disjoint subsets of $\mathcal{F}$, so our model captures a scenario where arriving peers are interested in multiple files.

Peers in Swarm C depart *immediately* upon retrieving all pieces in C. We partition peers into *types* according to (a) the swarm they belong to and (b) the set of pieces in their cache. Hence, a peer in Swarm C holding $S \subseteq \mathcal{F}$ is denoted to be of type $\langle C, S\rangle$. Assume the seed is of type $\langle\{\perp\}, \mathcal{F}\rangle$ for some piece $\perp \notin \mathcal{F}$. Denote $n_{\langle C,S\rangle}$ to be the number of type $\langle C, S\rangle$ peers and $\mathbf{n} = (n_{\langle C,S\rangle})$ to be the vector of numbers of peers in all types.

The seed uploads pieces at instants of a Poisson process of rate U. At each such instant, the seed contacts a peer selected uniformly at random among all peers across all swarms, and replicates a piece in $\mathcal{F}$ to this peer. Similarly, at instances that follow a Poisson process of rate $\mu > 0$, each peer contacts

another peer (also selected uniformly among all peers) and replicates a piece from its cache.

The piece replicated when a source (either a peer or the seed) contacts a receiver is determined by the source's piece selection policy. A broad class of work-conserving piece selection policies are considered, which satisfy the following:

Assumption 3.1.1 (Work-Conserving Piece Selection) *If a source in type $\langle C, S\rangle$ contacts a receiver in type $\langle C', S'\rangle$, (a) no piece is replicated if $S \subseteq S'$, otherwise (b) exactly one piece in $S \setminus S'$ is replicated, with piece $i \in S \setminus S'$ replicated as probability $h_{\langle C,S\rangle}(i, \langle C', S'\rangle, \mathbf{n})$, in $[0, 1]$, determined by the types of the source and the receiver, the piece id, and the current state $\mathbf{n}$. Function $h_{\langle C,S\rangle}$, also referred to as the piece selection policy, satisfies:*

$$\begin{aligned} \sum_{i \notin S \setminus S'} h_{\langle C,S\rangle}(i, \langle C', S'\rangle, \mathbf{n}) &= 0, \\ \sum_{i \in S \setminus S'} h_{\langle C,S\rangle}(i, \langle C', S'\rangle, \mathbf{n}) &= 1 \text{ if } S \not\subseteq S'. \end{aligned}$$

Suppose a type $\langle C, S\rangle$ source contacts a type $\langle C', S'\rangle$ receiver. Examples of work-conserving policies considered are:

Random Novel [RN]: If $S \setminus S' \neq \emptyset$, the source replicates a piece chosen uniformly from $S \setminus S'$.

Rarest First [RF]: Define the *availability* of a piece $i \in \mathcal{F}$ to be the number of peers holding it. The source replicates the piece in $S \setminus S'$ that has the least availability, with ties broken randomly.

Priority Rarest First [PRF]: The source prioritizes pieces within the swarm of the receiver: if $(S \setminus S') \cap C' \neq \emptyset$, it replicates the piece in $(S \setminus S') \cap C'$ that has the least availability; if $(S \setminus S') \cap C'$ is empty but $S \setminus S'$ is not, the source reverts to RF. **Priority Random Novel [PRN]** is defined similarly.

It is assumed that sources of the same type apply the same policy. The piece selection policy of the system is denoted by a tuple of $h_{\langle C,S\rangle}$ indexed by each $\langle C, S\rangle$, where all sources in type $\langle C, S\rangle$ apply the policy $h_{\langle C,S\rangle}$ in the tuple. Different policies h can co-exist across types: *e.g.*, the seed may implement RN, while peers implement PRF. Contrary to RN, the RF and PRF policies depend on the system state $\mathbf{n}$, and require knowledge of a global

property; as such, they are harder to implement in a distributed fashion. In a centralized setting, which includes most present BitTorrent implementations, the availability is monitored by a distinguished peer called the swarm tracker. Alternatively, distributed techniques such as gossiping or sampling can be used to obtain an estimate of the availability.

Under the assumption above, if no piece can be uploaded then the whole sampling interval is wasted, so each source can upload at most one piece per sampling interval. As that in Section 2.1, it is assumed that peer sampling and piece uploading are all instantaneous, which is a reasonable approximation if the time it takes for sampling and uploading is shorter than the length of intervals between sampling. Other notations are listed in the following.

Definition 3.1.1 *Let* $\mathcal{C} := \{C : C \in 2^{\mathcal{F}} \setminus \{\emptyset\}, \lambda_{\emptyset,C} > 0\}$ *be the set of swarms with positive arrival rates.*

Notice that type $\langle C, S\rangle$ *peers where* $C \subseteq S$ *does not exist because they can depart. Let* $\mathcal{T} := \{\langle C, S\rangle : C \in \mathcal{C}, S \in 2^{\mathcal{F}} \setminus \{\mathcal{F}\}, C \not\subseteq S\}$ *be the set of all types for peers existing in the network, and* $\tilde{\mathcal{T}} = \mathcal{T} \cup \{\langle\{\perp\}, \mathcal{F}\rangle\}$ *be the extended set containing the type of the seed. Define* $\mathcal{D} = \mathbb{N}^{|\mathcal{T}|}$ *to be the set of all possible vectors* $\mathbf{n}$*.*

Let $\lambda_{total} := \sum_{C:C\in\mathcal{C}} \lambda_{\emptyset,C}$ *be the total arrival rate. Let* $n_S := \sum_{C:C\in\mathcal{C}} n_{\langle C,S\rangle}$ *be the number of peers holding cache* S*. And let* $n := \sum_S n_S$ *be the total number of peers.*

The system evolution is described by a Markov process $\{\mathbf{n}(t)\}_{t\in\mathbb{R}_+}$ within state space $\mathcal{D}$. The transition rates of the process depend on how pieces are uploaded. Assume that the seed implements RN, and peers apply policies as in Assumption 3.1.1. Given a state $\mathbf{n}$, let $T_C(\mathbf{n})$ be the new state resulting from the arrival of a new peer in swarm C. Given $\langle C, S\rangle \in \mathcal{T}$ such that $i \notin S$, and a state $\mathbf{n}$ such that $n_{\langle C,S\rangle} \geq 1$, let $T_{\langle C,S\rangle,i}(\mathbf{n})$ denote the new state resulting from a type $\langle C, S\rangle$ peer downloading piece i. The positive entries of the generator matrix $Q = (q(\mathbf{n}, \mathbf{n}') : \mathbf{n}, \mathbf{n}' \in \mathcal{D})$ are given by:

$$q(\mathbf{n}, T_C(\mathbf{n})) = \lambda_{\emptyset,C}$$
$$q(\mathbf{n}, T_{\langle C,S\rangle,i}(\mathbf{n})) = \frac{n_{\langle C,S\rangle}}{n}\Big[\frac{U}{K-|S|} + \mu \sum_{v\in\mathcal{T}} n_v h_v(i, \langle C, S\rangle, \mathbf{n})\Big]$$

The main theorem regarding the stability region of multiple swarms is provided as Theorem 3.1.1, where the definitions of stability and instability

are provided in Definition 2.1.1.

Theorem 3.1.1 *If the seed implements RN and peers implement work-conserving piece selection policies in Assumption 3.1.1, the system is*

i) transient (unstable) if $\max_{i:i\in\mathcal{F}} \sum_{C:i\in C} \lambda_{\emptyset,C} > U$,

ii) positive recurrent and $E[n] < +\infty$ *in equilibrium (stable), if* $\max_{i:i\in\mathcal{F}} \sum_{C:i\in C} \lambda_{\emptyset,C} < U$.

An immediate corollary of Theorem 3.1.1, applied to the single swarm setup, is the stability region of Theorem 2.1.1. The theorem assumes that the seed uses RN, but numerical evaluations by simulations suggest that other work-conserving policies (RF) at the seed also exhibit the same stability region.

For comparison, consider an autonomous system with the same settings and assumptions stated in this section, but a) each peer is only allowed to sample targets uniformly from peers in its own swarm instead of over all swarms; and b) the seed allocates its capacity in a static way to all swarms: it maintains $|\mathcal{C}|$ independent Poisson clocks, each related to one swarm, such that when one clock ticks it uniformly samples a target from the swarm related to that clock; the ticking rate of each clock of the seed is fixed and the sum of all $|\mathcal{C}|$ rates is U. Theorem 2.1.1 directly applies to each swarm in the autonomous system and, thus, leads to the following:

Corollary 3.1.1 *Consider an autonomous multiswarm system. The seed can allocate its upload capacity so that the system is positive recurrent if* $\sum_{C\in\mathcal{C}} \lambda_{\emptyset,C} < U$*; the system is transient for all allocations of the seed's upload capacity if* $\sum_{C\in\mathcal{C}} \lambda_{\emptyset,C} > U$.

To better differentiate the system in Theorem 3.1.1 with the system in Corollary 3.1.1, name the former as a *universal system* or a multiswarm system in *universal mode*, and name the latter as an *autonomous system* or a multiswarm system in *autonomous mode*. Theorem 3.1.1 implies that universal systems yield a significant increase in the stability region comparing to autonomous systems. Observe that, when the files $C \in \mathcal{C}$ are disjoint, Theorem 3.1.1(ii) becomes $\max_{C\in\mathcal{C}} \lambda_{\emptyset,C} < U$. This defines a larger stability region than that given by Corollary 3.1.1. In particular, by bundling

swarms together, the stability region scales extremely well as the number of swarms increases: a single seed can support an unbounded number of swarms with constant arrival rate, with no effect on the stability region! However, bundling swarms together comes at the cost of increased delays. Hence, the number of swarms cannot be arbitrarily large in practice. In Section 3.5, we address this by proposing a hybrid system that, by alternating between the universal and autonomous mode, maintains the same stability region as a universal system while also ensuring small delays for large numbers of swarms.

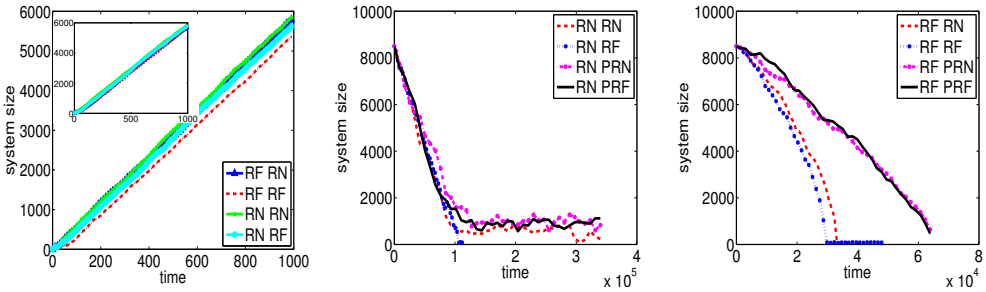
The stability region is insensitive to the piece selection policies implemented at peers. However, policies can be quite different w.r.t. other performance metrics. For example, as we show in Section 3.3, policies can differ drastically in how quickly they stabilize the system when operating within the stability region, as well as in how quickly the missing piece syndrome manifests when the system is not stable.

3.2 Validating the stability regions

In this section Theorem 3.1.1 and Corollary 3.1.1 are validated by simulation. We do so by studying the evolution of the system size (number of peers) n in autonomous and universal mode for a system comprising 3 swarms, each requesting a different 3-piece set. The seed rate is $U = 3.1$ and the arrival rate in each swarm is $\lambda = 3.0$; note that Theorem 3.1.1 and Corollary 3.1.1 imply that the autonomous mode is unstable while the universal system is stable, in this regime.

Figure 3.1(a) shows the evolution of the system size in autonomous mode, when the seed statically allocates 1/3 of its upload rate to each swarm, for different combinations of policies at the seed and the peers. All simulations start from an empty system. Even though applying RF at both the seed and the peers leads to a slightly smaller system size, the missing piece syndrome manifests in all four cases. We repeat these experiments with the seed allocating its rate dynamically, so that each swarm receives pieces at a rate proportional to its size. The results (inset of Figure 3.1(a)) show that instability persists in this setup too.

We repeat these experiments in universal mode, this time starting the



(a) Autonomous, static and dynamic (inset) allocation. (b) Universal, RN at seed. (c) Universal, RF at seed.

Figure 3.1: System size VS time. ("RN RF" means RN at seed and RF at peers, other legends follow similarly.)

system from an initial state comprising 8500 peers forming a *one-club*: all peers belong to the same swarm, and store in their cache all nine pieces *except for one common piece they request*. Figure 3.1(b) shows the system evolution when the seed applies RN; indeed the system stabilizes after 10^5 time units, confirming Theorem 3.1.1. The system stabilizes faster (in the order of 10^4 time units) when the seed applies RF, as seen in Figure 3.1(c), with RF at both seed and peers stabilizing the system the fastest (in roughly $3 \cdot 10^4$ time units). Interestingly, prioritizing pieces at peers (through either PRN or PRF) leads to *slower* stabilization: this is precisely because these policies reduce the diversity of pieces in the system.

3.3 Metastability of rarest first policy

Consider the same experiments as in Section 3.2 but set $U = 2.9$. As the arrival rate at each swarm is $\lambda = 3$, Theorem 3.1.1 stipulates that the system is unstable. A question here is how quickly the missing piece syndrome manifests in this case, depending on the piece selection policies.

To evaluate that, the following experiments are conducted. We start our simulations with initial system size n_0, where the initial state comprises all peers forming a one-club (*i.e.*, storing all pieces but one). We then terminate the simulation when either the system size increases to the threshold $\max(2000 + n_0, 2n_0)$ or the simulation time reaches 10^7, whichever occurs first. We first conduct this experiment with an empty initial state $n_0 = 0$; if

Table 3.1: Critical one-club, $U = 2.9$, 3 swarms each with arrival rate 3.0.

Policy		Critical	Final	Final One	Sim.
Seed	Peer	n_0	Size	Club Ratio	Duration
RN	PRN	0	2000	95.6%	13181
	PRF	0	2000	95.4%	17211
	RN	0	2000	93.3%	13603
	RF	500	2500	94.7%	22655
RF	PRN	2100	4200	98.1%	74655
	PRF	2000	4000	98.0%	51415
	RN	8000	16000	99.4%	283197
	RF	8100	16200	99.4%	323738

the experiment does not reach the threshold in 10^7 units, we increase n_0 by 100 and repeat the experiment. This way, we identify the *critical one-club size*: if the system reaches a state with a one-club of that size, it becomes unstable.

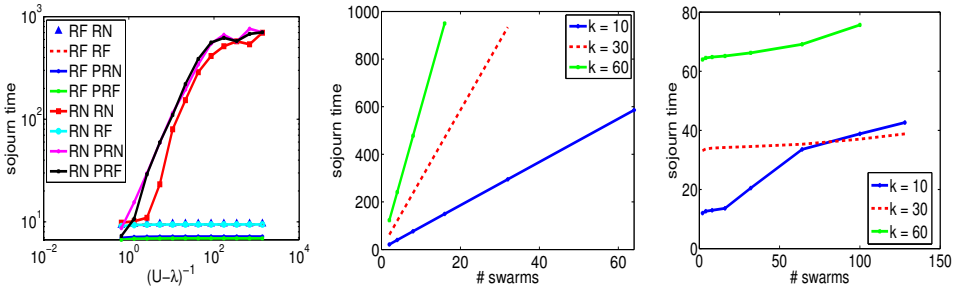
Our simulation results for the case where the seed applies RN are summarized in the top half of Table 3.1. We see that the missing piece syndrome indeed manifests at the critical initial conditions, with the one-club comprising more than 90% of the peer population at termination time. When peers use any policy other than RF, the critical one-club size is 0. In contrast, when peers use RF, the syndrome manifests only when $n_0 = 500$; indeed, using RF improves the diversity of pieces in the system, which in turns makes reaching a critical one-club size more difficult. This behavior becomes even more striking when the seed uses RF: as shown in the bottom half of Table 3.1, piece diversity is so high that critical one-club sizes lie between 2 and 8 thousand peers.

Crucially, in all simulations starting from an initial size below the critical value, we observe interestingly that the system size actually *decreases* to a size below 200 and lingers around this value for the entire 10^7 time units! This implies that, though the system is clearly not stable in any of the cases in Table 3.1, applying RF at the seed or peers yields *metastability*: although there exists a critical one-club size, its value is so high that it is quite hard to reach from the "typical" size at which the system operates most of the time ($\sim$200 peers in our simulations).

A natural question to ask is what is the critical value n_0, as well as what is the "typical" size at which a metastable system operates most of the time;

we revisit these questions, in Section 3.5.

3.4 Average sojourn time



(a) Approach boundary from below. (b) Universal, $U = 3.0$, $\lambda = 1.5$. (c) Hybrid, $U = 3.0$, $\lambda = 1.5$.

Figure 3.2: Average sojourn time for universal swarms.

In this section our attention is turned to the sojourn time. First we study a universal system comprising 3 swarms with 3 pieces each. The seed rate is fixed at $U = 3.0$ and the swarm arrival rate varies as $\lambda = U(1 - \frac{1}{2^i})$, for $i = 1, \ldots, 10$, remaining within the stability region but approaching U from below. In Figure 3.2(a), we plot the average sojourn time for different piece selection policies as a function of $1/(U - \lambda)$. We observe that, as λ approaches U, the sojourn time under the RN policy at the seed increases considerably, with the exception of the RN-RF case, *i.e.*, when peers use RF. In all four cases for which the seed uses RF, the sojourn time remains practically constant as λ approaches U. This is consistent with the fact that, by metastability, when the seed uses RF the system size remains small most of the time even if $\lambda > U$; as such, there is no sharp increase in the sojourn time as λ approaches U from below.

We next study how the sojourn time scales with the number of swarms L. In Figure 3.2(b), we plot the average sojourn time vs. L for the case where each swarm comprises peers requesting a k-piece file, for $k \in \{10, 30, 60\}$. Note that the total number of pieces is $K = kL$. We observe that across all values of k, the average sojourn time increases linearly as L increases. Similarly, the sojourn time also increases proportionally to k. Thus, the increased stability offered by bundling swarms together comes at the cost of

increased delays; we address this in the next section by showing that delays can be suppressed for a wide range of values of L by using a *hybrid* approach, alternating between universal and autonomous mode.

3.5 Stable, low sojourn universal swarms

Our simulations suggest that, in a metastable swarm, there are two important system sizes: the *operating* size $n_{\rm op}$, which is the size around which the system stays most of the time, and the *critical size* n_0, which is the size of a one-club that, once attained, leads the system to instability. If the two sizes are sufficiently far apart from each other, the system will exhibit metastability: when $N \approx n_{\rm op}$, it will take a long time for N to reach n_0, from which the missing piece syndrome manifests.

Calculating exactly $n_{\rm op}$ and n_0 is quite challenging. Nevertheless, we can derive some simple estimates of $n_{\rm op}$ and n_0 when (a) the system comprises of a single swarm, (b) $\lambda > U$, and (c) both the seed and peers use RF.

Consider a single swarm, where peers arrive with rate λ and wish to download K pieces. Assume that the seed has upload rate $U < \lambda$, so that the system is unstable, and peers have upload rate μ. When the system is in the operating state, we expect that the diversity of pieces is high enough, so that every contact a peer makes leads to a piece download. Under this assumption, as a peer wishes to download K pieces, the expected sojourn time is K/μ. By Little's law, as the arrival rate is λ, an approximation of the operating size is therefore $n_{\rm op} \approx \frac{\lambda K}{\mu}$. Estimating the critical size of a one-club requires a more involved argument. Suppose that a one-club with size B has formed. For B large, peers outside the one-club download pieces from the one-club at a rate close to μ. As such, the expected time it takes a new peer to be converted to a one-club peer is approximately $(K-1)/\mu$; hence, the number of young peers, *i.e.*, peers outside the one-club, is approximately $\lambda(K-1)/\mu$. Young peers can become infected, *i.e.*, obtain the piece the one-club peers are missing. As the seed uses the RF policy, the rate with which the seed infects young peers is $U\frac{\lambda(K-1)/\mu}{B+\lambda(K-1)/\mu}$. Ignoring the fact that young peers may also infect other young peers, and assuming that an infection occurs at an instant sampled uniformly at random within a young peer's lifetime, each infected peer stays for $(K-1)/2\mu$ time units before it departs, in expectation.

Then, the drift of the one-club is roughly $\Delta B = \lambda - U - U\frac{\lambda(K-1)/\mu}{B+\lambda(K-1)/\mu}\frac{(K-1)}{2\mu}\mu$. Requiring the drift to be zero and solving for B, we obtain that in such a single swarm system:

$$n_{\text{op}} \approx \frac{\lambda K}{\mu}, \text{ and } n_0 \approx \frac{\lambda(K-1)}{\mu}[\frac{1}{2}\frac{U(K-1)}{\lambda - U} - 1]. \tag{3.1}$$

Using these two estimates, we propose a *hybrid system* that attains the increased stability region of the universal swarm, while also ensuring that the sojourn times remain small for a wide range values of L. The hybrid system alternates between the autonomous mode, whereby swarms operate in isolation while sharing a U/L portion of the seed's capacity, and the universal mode, where swarms are bundled together. In particular, consider a system with L swarms, each requesting a file of $k = K/L$ pieces. The system alternates between the two modes according to the following rules: (a) If in autonomous mode, the system switches to universal mode if any single swarm has size $\geq n_{\text{op}} + \max(n_0, 2n_{\text{op}})$; (b) if in universal mode, the system switches to autonomous mode if each piece requested by a swarm is held by at least $\max(n_{\text{op}}/10, 1)$ peers within the swarm. Values n_{op}, n_0 are computed by (3.1), assuming an upload rate U/L and a number of pieces k. Intuitively, the universal mode is applied when there is strong evidence that the missing piece syndrome is manifesting, as the swarm size becomes greater than $n_{\text{op}} + n_0$. The system reverts to an autonomous mode when there is enough diversity in each swarm—each piece is held by at least the one tenth of the peer population.

The hybrid system reverts to a universal system if system size keeps large, so it exhibits the increased stability region of universal swarms. Figure 3.2(c) shows the sojourn time of a hybrid system as L increases. In contrast to Figure 3.2(b), for $k = 30$ and $k = 60$, the sojourn time stays close to the value attained when $L = 1$ ($\sim$33 and $\sim$64 time units, respectively). For $k = 10$, the sojourn time starts increasing linearly after $L = 12$.

These improved sojourn times appear precisely because of the metastability. Indeed, swarms operate fine most of the time without the intervention of other swarms, and this is why they experience the same delay as if $L = 1$. As $U/L < \lambda$, the autonomous mode is unstable; however, on the few (and rare) occasions when the missing piece syndrome manifests, bundling swarms together ensures the system quickly stabilizes and reverts to its operating size.

The knee observed for $k = 10$ suggests that this behavior cannot be sustained for arbitrarily large L. Equation (3.1) can help us give an approximate answer to how large L can be. Indeed, for the system to be metastable, the critical one-club size must be significantly larger than the operating size. Requiring that $n_0 > 2n_{\text{op}}$, so that the missing piece syndrome rarely manifests, and taking $K/(K-1) \approx 1$, gives the following heuristic for metastability when $L = 1$: $K\frac{U}{\lambda - U} > 6$. Consider now $L > 1$ swarms in autonomous mode, each requesting $k = K/L$ pieces. Each swarm gets a U/L upload rate in autonomous mode. Then, the above condition becomes $L < \frac{U}{6(\lambda - \frac{U}{L})}k \approx \frac{U}{6\lambda}k$. In other words, the hybrid system can support a number of swarms L with small delay so long as L is of the order of k, the number of pieces in each swarm. As the number of pieces in a file typically numbers in the thousands, this implies that the above system can sustain low sojourn times for a large number of swarms.

3.6 Proof of transience

In this section the detailed proof of the transience, i.e. instability, in Theorem 3.1.1 is provided. If $K = |\mathcal{F}| = 1$, Theorem 3.1.1 follows because $\mathbf{n}$ is an $M/M/1$ queue, with arrival rate $\lambda_{\emptyset,\mathcal{F}}$ and service rate U.

Assume that $K \geq 2$ and that, *w.l.o.g.*, $\lambda_1 := \sum_{C:1\in C} \lambda_{\emptyset,C} > U$. Notice that $\mathbf{n}$ is irreducible; as such, to show it is transient, it suffices to show that there exists a transient state [54, Proposition 6.3.5]. We show that the initial state where many peers are missing piece one is a transient state, beginning from which the number of peers will converge to infinity with a positive probability. Transience directly implies that the number of peers converges to infinity with probability one, as (a) there is a finite number of states where the number of peers is bounded by a constant, and (b) the probability for $\mathbf{n}$ to stay in any finite set is zero. To construct a transient initial state, we assume:

Assumption 3.6.1 *Select positive values* $\epsilon, \xi, \rho, \epsilon_o, B, N_o$ *so that*

$$3\epsilon < \lambda_1 - U, \xi < 0.5, \epsilon > 4K\xi U, \tag{3.2}$$

$$\rho := 2\xi(K-1) < 0.5, \epsilon_o < \xi(\lambda_1 - U - 3\epsilon), \tag{3.3}$$

$$e^{\lambda_{total}[2(K-1)/\mu+1]}2^{-B} \leq 0.1(1-2^{-\epsilon_o}), \tag{3.4}$$

$$64K^2\xi U \leq 0.2B(\epsilon - 4K\xi U), \tag{3.5}$$

$$\lambda_{total} \leq 0.2B\epsilon, U \leq 0.2B\epsilon, \tag{3.6}$$

$$N_o > 6B, B+1 < \xi(N_o - 3B - 1). \tag{3.7}$$

We partition the set of peers in two classes: *one-club peers*, which are the peers having all pieces in $\mathcal{F} \setminus \{1\}$, and *young peers*, which are peers missing at least one piece from $\mathcal{F} \setminus \{1\}$. We also refer to peers holding piece 1 as *infected peers*; note that infected peers are necessarily young.

Definition 3.6.1 *Define the following random processes:*

- A_t : *cumulative number of arrivals of peers wanting to download piece one, up to time t*
- N_t : *number of peers at time t*
- Y_t : *number of young peers at time t*
- D_t : *cumulative number of uploads of piece one by infected peers, up to time t*
- Z_t : *cumulative number of uploads of piece one by the seed up to time t*

We construct the following initial state:

Assumption 3.6.2 *At* $t = 0$, $N = N_o$ *and all* N *peers are one-club peers.*

Let τ be the extended stopping time defined by $\tau = \min\{t \geq 0 : Y_t \geq \xi N_t\}$, with the usual convention that $\tau = \infty$ if $Y_t < \xi N_t$ for all t.

Lemma 3.6.1 *Under assumptions 3.6.1 and 3.6.2,*

$$P\{A_t > -B + (\lambda_1 - \epsilon)t \ \ \textit{for all } t \in [0,\tau)\} \geq 0.9, \tag{3.8}$$

$$P\{Z_t < B + (U+\epsilon)t \ \ \textit{for all } t \in [0,\tau)\} \geq 0.9, \tag{3.9}$$

$$P\{Y_t < B + \epsilon_o t \ \ \textit{for all } t \in [0,\tau)\} \geq 0.9, \tag{3.10}$$

$$P\{D_t < B + \epsilon t \ \ \textit{for all } t \in [0,\tau)\} \geq 0.9. \tag{3.11}$$

The proof of Lemma 3.6.1 is given after this section. Lemma 3.6.1 implies Lemma 3.6.2; thereby Theorem 3.1.1(i) follows because the state in assumption 3.6.2 is a transient state.

Lemma 3.6.2 $P\{\tau = \infty \quad and \quad \lim_{t\to\infty} N_t = +\infty\} \geq 0.6$

Proof. Let $\mathcal{Z}$ be the intersection of the four events on the left sides of (3.8)-(3.11). We have $P(\mathcal{Z}) \geq 0.6$. Note that $N_0 + A_t - D_t - Z_t$ is no larger than the number of peers wanting to download piece one at t. So, on $\mathcal{Z}$,

$$N_t \geq N_0 + A_t - D_t - Z_t > N_0 - 3B + (\lambda_1 - U - 3\epsilon)t$$

for all $t \in [0, \tau)$. We claim that τ cannot be finite on $\mathcal{Z}$. Otherwise, at time τ,

$$\begin{aligned} \xi &\leq \frac{Y_\tau}{N_\tau} < \frac{B + 1 + \epsilon_o \tau}{N_0 - 3B - 1 + (\lambda_1 - U - 3\epsilon)\tau} \\ &\leq \max\left\{\frac{B+1}{N_0 - 3B - 1}, \frac{\epsilon_o}{\lambda_1 - U - 3\epsilon}\right\} < \xi. \end{aligned}$$

Contradiction!! Thus, on $\mathcal{Z}$ we have $\tau = \infty$. So $N_t \to \infty$, and $\forall t$, $Y_t/N_t < \xi$. Thus, $\mathcal{Z}$ is a subset of the event on the left side of Lemma 3.6.2, and so Lemma 3.6.2 follows. □

To prove Theorem 3.1.1(i), it remains to prove Lemma 3.6.1, which is provided in the following.

3.6.1 Proof of Lemma 3.6.1

Define an alternative process to be a process the same as the original process $\mathbf{n}$, but it terminates at time $\tau = \min\{t \geq 0 : Y_t \geq \xi N_t\}$. It is sufficient to prove for the alternative process. In the alternative process, A is a Poisson process with rate λ_1, and Z is stochastically dominated by a Poisson process with rate U. Thus, both (3.8) and (3.9) follow from (3.6) and the consequence of Kingman's moment bound in Proposition A.4.1.

The inequality (3.10) follows from (3.4), Lemmas 3.6.3 and A.6.1:

Lemma 3.6.3 *The process Y is stochastically dominated by the number of customers in one $M/GI/\infty$ queue initially empty, arrival rate λ_{total}, and service times $\sim Gamma(K-1, 2/\mu)$.*

Table 3.2: Specification of comparison system

Alternative system	Comparison system
The seed creates infected peers at a rate $\leq \xi U$.	The seed creates infected peers at rate ξU.
An infected peer creates infected peers at rate $\leq \xi\mu$.	An infected peer creates infected peers at rate $\xi\mu$.
An infected peer uploads piece one to one-club at a rate $\leq \mu$.	An infected peer uploads piece one to one-club at rate μ.
Any infected peer needs at most $K-1$ additional pieces, at rate $\geq \mu/2$.	A new infected peer must get $K-1$ additional pieces, at rate $\mu/2$.

To prove (3.11), consider the stochastic system described in Table 3.2, which we call the *comparison system*. It should be clear to the reader that both the alternative system and the comparison system can be constructed on the same underlying probability space; in particular, such a construction can be done so that any infected peer in the alternative system at a given time is also in the comparison system. To enforce this, when a peer becomes infected in the alternative system, we require that (a) it also arrives to the comparison system, (b) it discards all pieces it may have downloaded before becoming infected, and (c) it subsequently ignores all opportunities to download except those occurring at the times its internal Poisson clock with ticking rate $\mu/2$ ticks. Because infected young peers may stay longer in the comparison system than in the alternative system, some of the peers in the comparison system correspond to peers that already departed from the alternative system. There can also be some infected peers in the comparison system that never existed in the alternative system because of the higher arrival rate of infected peers in the comparison system.

In all cases, any infected peer in the alternative system is also in the comparison system. That is, any of the following events occurring in the alternative system also occurs in the comparison system: (a) the seed creates an infected peer, (b) an infected peer creates an infected peer, and (c) an infected peer replicates piece one to a one-club peer. Events of types (b) and (c) correspond to the two possible ways that infected peers can upload piece one. Therefore, this property implies the Lemma 3.6.4, where $\widehat{D}$ is the cumulative number of uploads of piece one by infected peers, up to time t, in the comparison system.

Lemma 3.6.4 *The process* $(D_t : t \geq 0)$ *is stochastically dominated by* $(\widehat{D}_t : t \geq 0)$.

Lemma 3.6.5 $(\widehat{D}_t : t \geq 0)$ *can be stochastically dominated by a compound Poisson process* $(\widetilde{D}_t : t \geq 0)$*, with arrival rate of batches* $= \xi U$*, and first and second moments of batch sizes bounded by* $4K$ *and* $64K^2$*, respectively*

Hence, (3.11) with D replaced by $\widetilde{D}$ follows from Proposition A.4.1 and (3.5). Lemma 3.6.1 therefore follows.

3.7 Proof of positive recurrence

In this section the proof of positive recurrence, i.e., stability, in Theorem 3.1.1 is provided. We establish the stability by the Foster's criterion [57]. Specifically, we use the result in Lemma 2.5.1, which is a standard version of the criterion. First the description of a Lyapunov function is provided, then the function is shown to satisfy the above Foster criterion. The following definitions are provided for the proof in this section only.

Definition 3.7.1 $\Delta := U - \max_{i:i\in\mathcal{F}} \sum_{C:i\in C} \lambda_C > 0$.

Definition 3.7.2 $\mathcal{E}_C := \{\langle C', S'\rangle : C' \not\subseteq C, S' \subseteq C\}$ *is the set of types of peers which may hold the set of pieces* C *in the future,* $E_C := \sum_{\langle C',S'\rangle\in\mathcal{E}_C} n_{\langle C',S'\rangle}$ *is the number of peers with types in* $\mathcal{E}_C$*:*

Definition 3.7.3 $\mathcal{H}_C := \bigcup_{i\in\mathcal{F}\setminus C} \{\langle C', C\cup\{i\}\rangle : C' \not\subseteq C\cup\{i\}\}$ *is the set of types of peers which hold one more piece than* C*,* $H_C := \sum_{\langle C',S'\rangle\in\mathcal{H}_C} n_{\langle C',S'\rangle}$ *is the number of peers with types in* $\mathcal{H}_C$*.*

Definition 3.7.4 $r \in (0, \frac{1}{2})$*,* $d \in (1, \infty)$*,* $\beta \in (0, \frac{1}{2})$*,* $\alpha := 8K(\lambda_{total} + \Delta/2)/U$ *are four constants, and*

$$\phi(x) := \begin{cases} (2d + \frac{1}{2\beta} - x) & 0 \leq x \leq 2d \\ \frac{\beta}{2}(x - 2d - \frac{1}{\beta})^2 & 2d < x \leq 2d + \frac{1}{\beta} \\ 0 & x > 2d + \frac{1}{\beta} \end{cases}$$

Definition 3.7.5 *The Lyapunov function W is defined as $W := \sum_C r^{|C|} T_C$, where*

$$T_C := \begin{cases} \frac{1}{2}E_C^2 + \alpha E_C \phi(H_C), |C| \leq K-2, \\ \frac{1}{2}E_C^2, |C| = K-1 \end{cases}$$

As will appear, function ϕ is affine in some range, and thus the Lyapunov function W is quadratic in the state variable $\mathbf{n}$ in a corresponding range. However we need to go beyond quadratic functions in order to apply the Foster criterion.

In the following, it is shown that for suitable choices of constants r, d, β the function W satisfies Foster's criterion. Notice that the framework, notations and definitions of the proof in the following are quite similar to those of the proof in Section 2.5, but detailed calculations and parameters are different. In order to stay organized and be convenient for readers, we provide the full details below instead of referring back to Section 2.5 for similar notations and definitions.

Definition 3.7.6 *$M_\phi := 3d + \frac{1}{\beta}$. Note $M_\phi > \max_x \phi(x)$ and $M_\phi > \min\{x : \phi(x) = 0\} + d > 1$.*

Definition 3.7.7 *Define $D_{\langle C,S\rangle}$ to be the transition rate for type $\langle C, S\rangle$ peers to download pieces.*

Define $D_S := \sum_{C:C\in\mathcal{C},C\not\subseteq S} D_{\langle C,S\rangle}$.

Define $D_{total} := \sum_{S:S\subsetneq\mathcal{F}} D_S$.

Note that $D_{\langle C,S\rangle}, D_S, D_{total}$ are functions of the state $\mathbf{n}$.

Definition 3.7.8 *Define Γ_{J_1,J_2} for $J_1, J_2 \in \mathcal{T}, J_1 \neq J_2$ to be the transition rate for type J_1 peers to become type J_2 peers.*

Define $\Gamma_{\mathcal{X}_1,\mathcal{X}_2} := \sum_{J_1:J_1\in\mathcal{X}_1} \sum_{J_2:J_2\in\mathcal{X}_1} \Gamma_{J_1,J_2}$, where $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$.

Define $D_{\mathcal{H}_S} := \sum_{J:J\in\mathcal{H}_S} D_J$, for $\mathcal{H}_S$ in Definition 3.7.3.

Define $A_{\mathcal{H}_S}$ to be the total transition rate for peers to join the group of peers with types in $\mathcal{H}_S$.

Let $\mathcal{P}_S := \{\langle C, S\rangle : C \not\subseteq S, C \in \mathcal{C}\}$ be the set of types of peers holding piece set S.

Notice that Γ, D, A are all functions of $\mathbf{n}$.

Lemmas 3.7.1 and 3.7.2 obviously hold:

Lemma 3.7.1 *Function ϕ verifies $\phi'(x) = -1$ for $0 \leq x \leq 2d$ and $\phi'(x) = 0$ for $x \geq 2d + 1/\beta$. And ϕ' increases linearly from -1 to 0 in $[2d, 2d + 1/\beta]$.* $\forall x \geq 0, \phi'(x) \in [-1, 0]$.

Lemma 3.7.2 $D_S \leq U + \mu \min\{n_S, n - n_S\}$, $D_{total} \leq U + n\mu$, $D_S \geq (U + H_S\mu)\, n_S/n$.

In the proof, we consider the following two classes of states, where σ is to be selected within $\sigma \in (0, 1/2)$. The classes overlap and their union includes every non-zero state:

Definition 3.7.9 *Class I is the set of states* $\mathbf{n}$ *such that there exists $S \subsetneq \mathcal{F}$, and $n_S/n > 1 - \sigma$; Class II is the set of states* $\mathbf{n}$ *such that there exist $C_1, C_2 \subsetneq \mathcal{F}$, so that, $n_{C_1}/n > \sigma/2^K$ and $n_{C_2}/n > \sigma/2^K$.*

For a specific σ, a state $\mathbf{n}$ can either be Class I or Class II, or both. The main idea of the proof is to show that W is a valid Lyapunov function for an appropriate choice of (r, d, β, σ). The given parameters of the network, $\lambda_C (C \in \mathcal{C}), U$ and μ, are treated as constants. Functions on the state space may or may not depend on the variables r, d, β and σ. It is convenient to adopt the big theta notation $\Theta(\cdot)$, with the understanding that it is uniform in these variables; this is summarized in the following definitions.

Definition 3.7.10 *Given functions f and g on the state space $\mathcal{D}$, we say $f = \Theta(g)$ if there exist $k_1, k_2, n_0 > 0$, not dependent on (r, d, β, σ), such that $k_1|g(\mathbf{n})| \leq |f(\mathbf{n})| \leq k_2|g(\mathbf{n})|$ for all $\mathbf{n}$ such that $n > n_0$.*

For example, $2 = \Theta(1), \lambda_{total} n = \Theta(n)$, $d = \Theta(d)$, $1 \leq \Theta(n)$ and $\Theta(n) - \Theta(n)/2 = \Theta(n)$. Notice that d and $\Theta(1)$ cannot be compared. Similarly, we adopt notions of "small enough" and "large enough" that are uniform in (r, d, β, σ):

Definition 3.7.11 *We say that "condition A is true if $x > 0$ is* small enough*" if there exists a constant $k > 0$, not depending on (r, d, β, σ), such that A is true for any $x \in (0, k)$. Similarly, we say that "condition A is true if $x > 0$ is* large enough*" if there exists a constant $k > 0$, not depending on (r, d, β, σ), such that A is true for any $x \in (k, \infty)$.*

We identify an approximation to the drift of W. Notice that the infinitesimal generator Q is linear, so that $Q(W) = \sum_C r^{|C|} Q(T_C)$, with $Q(T_C) = \frac{1}{2} Q(E_C^2) + \alpha Q(E_C \phi(H_C))$.

Definition 3.7.12 *Define $\mathcal{Q}W$, an approximation of $Q(W)$, as $\mathcal{Q}W := \sum_C r^{|C|}\mathcal{Q}T_C$, with*
$\mathcal{Q}T_C := E_C Q(E_C) + \alpha E_C Q(\phi(H_C))$ *if* $|C| \le K-2$,
$\mathcal{Q}T_C := E_C Q(E_C)$ *if* $|C| = K-1$.

The proof relies on Lemmas 3.7.3 to 3.7.6, which are conditioned on (r, d, β, σ), to bound terms in $Q(W)$ and $\mathcal{Q}W$. The proofs of these lemmas are provided in Appendix C.

Lemma 3.7.3 *Bound for the approximation error:*

$$|Q(W) - \mathcal{Q}W| \le M_\phi(D_{total}+1)\Theta(1). \tag{3.12}$$

Lemma 3.7.4 *If d is large enough, $\forall C \subsetneq \mathcal{F}$, $Q(E_C) \le \Theta(1)$, $Q(\phi(H_C)) \le M_\phi\Theta(1)$, and $\mathcal{Q}T_C \le M_\phi\Theta(E_C) \le M_\phi\Theta(n)$.*

Lemma 3.7.5 *If d is large enough, $\sigma M_\phi, \beta$ are small enough, for any $S \subsetneq \mathcal{F}$ and any nonzero state $\mathbf{n}$ such that $n_S/n > 1-\sigma$, $\mathcal{Q}T_S \le -\frac{1}{2}\Delta E_S$.*

Lemmas 3.7.4 and 3.7.5 imply Lemma 3.7.6:

Lemma 3.7.6 *If d is large enough, $\beta, rM_\phi, \sigma M_\phi r^{-K}$ are small enough,*

(a) on Class I, $\mathcal{Q}W \le -r^K\Theta(n)$;

(b) on Class II, $\mathcal{Q}W \le -r^K\sigma^3\Theta(n^2) + M_\phi\Theta(n)$.

Notice that the conditions of Lemmas 3.7.3 to 3.7.6 are consistent with each other, so Lemma 3.7.7 is claimed.

Lemma 3.7.7 *There exists (r, d, β, σ) satisfying all conditions of Lemmas 3.7.3 to 3.7.6, such that W is a valid Lyapunov function.*

Proof. On Class I, $D_{total} = D_S + \sum_{C:C\neq S} D_C \le 2U + 2(n-n_S)\mu \le 2U + 2\sigma n\mu = \Theta(1) + 2\sigma\Theta(n)$. So Lemma 3.7.3 implies that on Class I, $|Q(W) - \mathcal{Q}W| \le \sigma M_\phi\Theta(n) + M_\phi\Theta(1)$. Combined with Lemma 3.7.6(a), on Class I, $Q(W) \le -r^K\Theta(n) + M_\phi\Theta(1)$ if $\sigma M_\phi r^{-K}$ is small enough. On Class II, $D_{total} \le U + n\mu = \Theta(n)$, so Lemma 3.7.3 implies that $|Q(W) - \mathcal{Q}W| \le M_\phi\Theta(n)$. On Class II, Combined with Lemma 3.7.6(b), $Q(W) \le -r^K\sigma^3\Theta(n^2) + M_\phi\Theta(n)$. Thus, if (r, d, β, σ) satisfies conditions of Lemmas 3.7.3 to 3.7.6, and $\sigma M_\phi r^{-K}$ is small enough, $\exists \varsigma > 0$ such that $Q(W) \le -\varsigma n$ whenever n is larger than some constant. Thus W is a Lyapunov function. □

Chapter 4

FLOW LEVEL STREAMING

4.1 Problem setup and model

Consider a network containing one server and N peers (nodes), labeled as $1, 2, ...N$. One video to be broadcast from the server to all nodes is cut into M substreams. Each substream is transmitted through a directed broadcast tree. We consider the problem of how to build broadcast trees, so as to avoid interference, achieve coverage and reduce delay. *For convenience we refer to nodes as peers in this chapter, interchangeably.*

Consider a flow level model. Let V denote the set of N nodes. As illustrated in Figure 4.1, assume there are M root nodes (or roots, $M << N$) in V, each of which always has an incoming link from the server, and always receives a distinct substream via such link. Each root works as an "agent" of the server to further distribute its received substream. Let $\mathcal{R}$ denote the set of roots. For convenience, assume the root nodes are nodes 1 through M, and label each substream by the label of the root receiving the stream from the server.

Assume nodes in V can randomly contact each other and build directed links among themselves. Let E denote the set of all such links. Through each link one and only one substream can be transmitted from the tail to the head of the link. Assume that at the time a link is built, the substream

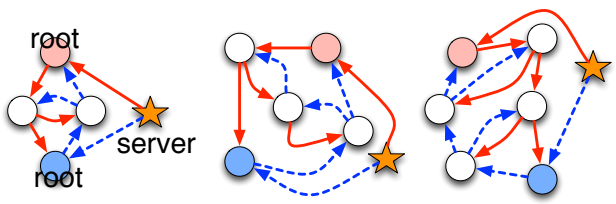


Figure 4.1: Spanning trees with $\lceil \log N \rceil$ depth, $N = 4, 5, 6$.

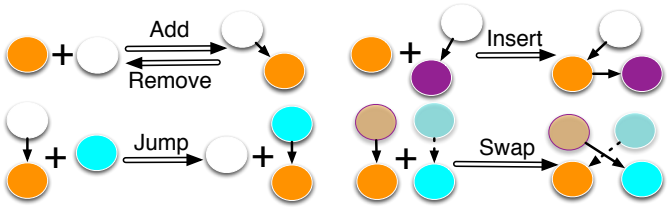


Figure 4.2: All five types of link updates. Link colors in the "Swap" transform may or may not be the same.

to be transmitted on it is also determined. Assume each link is colored by the label of the substream transmitted on it. Let E_i denote the set of all links with color i for $i \in \mathcal{R}$. The set of all E_i's is a partition of E. *A node u can receive substream i if and only if in graph (V, E_i) there is a directed path from root i to u; and the delay of receiving substream i is modeled by the number of hops of the shortest path from i to u.* Let V_i denote the set of nodes to which there exists a path from root i in (V, E_i). That is, V_i is the set of nodes which can receive substream i.

Assume that to recover the video, a node u must receive at least K out of the M substreams: $|\{i \in \mathcal{R} : u \in V_i\}| \geq K$. It is possible that $K < M$, corresponding to the use of source coding. Assume each node has a constraint on upload capacity (outdegree): node u can build at most $\bar{d}_u$ outgoing links, whatever their colors are. Figure 4.1 contains examples of (V, E) for $M = K = 2$ and $N = 4, 5, 6$, where roots have outdegree one and other nodes have outdegree two. In Figure 4.1, for each root i, (V, E_i) is a spanning tree with the minimum depth under outdegree constraint, with the tree depth defined as the maximum number of hops over all root-leaf paths.

Five types of link updates are considered, as shown in Figure 4.2. Each link update can be executed locally because at most four nodes are involved. Notice that link updates in Figure 4.2 are just combinations of building a link and removing a link. Any node on the left side of a link update in Figure 4.2 can initiate that update by exchanging messages with other nodes involved.

Assume each node maintains a Poisson clock which ticks at rate $\mu = 1$, independently of Poisson clocks of other nodes. Whenever the clock of a node ticks, the node samples a target chosen from among the other nodes uniformly at random, and decides whether to execute link updates in Figure 4.2 or not. *In this thesis, we assume link updates happen instantaneously and at most one update can be executed at each sampling.* It makes the problem tractable and

is also a reasonable relaxation because one link update consists of building at most two links and removing at most two links. Here we normalize the time so $\mu = 1$.

In Figure 4.1, the spanning tree for each substream has depth $\lceil \log N \rceil$. In the next section, we show that our algorithm insures that all substreams can be broadcasted through distinct trees with $\log N + O(1)$ depths and each node can receive enough substreams. The following notation is used:

- $L_i(u)$: the *depth to i of u, defined as the minimum number of hops from root $i \in \mathcal{R}$ to u in (V, E_i). Define $L_i(u) = +\infty$ if no path exists from i to u in (V, E_i).*
- l_i^u : *the depth to i buffered by u, which is continually updated.*
- V_i : *for each $i \in \mathcal{R}, V_i := \{u : u \in V, L_i(u) < +\infty\}$.*
- *i-link: an i-link refers to a link colored i.*
- $d_i(u)$: *the number of outgoing i-links of node u. Let $d(u) = \sum_{i \in \mathcal{R}} d_i(u)$ be the total number of outgoing links of u.*
- *i-parent: u is an i-parent of v if $(u, v) \in E_i$.*
- *i-child: v is an i-child of u if $(u, v) \in E_i$.*
- *i-internal: Node u is an i-internal node if $L_i(u) + 2 \leq \max_{w \in V_i} L_i(w)$.*
- *i-leaf: Node u is an i-leaf if $d_i(u) = 0$.*
- *available: Node u is* available if u has an incoming link and $d(u) < \bar{d}_u$.
- mixed node: A node is mixed if it has at least two outgoing links with different colors.
- mixed-i-j-node: Node u is a mixed-i-j-node if $d_i(u)d_j(u) > 0$.
- Write $(a_i) < (b_i)$ for two sequences to mean $\forall i, a_i < b_i$.

4.2 A distributed algorithm for tree management

Our main algorithm is summarized in Section 4.2.5 as Procedure CombinedUpdate, which is a topology update procedure to be run after nodes randomly contact each other. In the following we introduce CombinedUpdate by parts. Notice that by an assumption stated in Section 4.1, running of CombinedUpdate is instantaneous. Assume each node knows its parents, children, and colors of its incoming and outgoing links. Each node buffers its depths to all roots. To begin, we assume the initial graph (V, E) satisfies Assumption 4.2.1.

Assumption 4.2.1 (Tree Initially)
At time 0, (V, E) *satisfies the following:*

1. *Each node* u *has at most* K *incoming links, has at most one incoming* i*-link for each* $i \in \mathcal{R}$*, and at most* $\bar{d}_u$ *outgoing links, which implies that*

2. (V_i, E_i) *is a directed tree rooted at* i *for each* $i \in \mathcal{R}$*.*

Notice that 1) implies 2) in Assumption 4.2.1. And notice that cycles may appear in $(V \setminus V_i, E_i)$ even if (V_i, E_i) is a tree, as shown in Figure 4.4. Assumption 4.2.1 can be easily satisfied by requiring all nodes to remove extra incoming or outgoing links at time 0. We will show that the properties in Assumption 4.2.1 are preserved by the update procedures we shall define. In the following the procedure for depth update is discussed first.

4.2.1 Distributed depth update

Each node buffers its depths to each root $i \in \mathcal{R}$. Notice l_i^u denotes the depth to i buffered by node u.

Procedure DepthUpdate (Node u, Color i)

If node u is root i, u sets l_i^u to 0; otherwise,

- if u has no incoming i-links, u sets l_i^u to $+\infty$;
- if u has incoming i-links, u sets l_i^u to $l + 1$, where l is the minimum depth to i buffered by all i-parents of u.

Figure 4.3: Greedy Tree Cover.

We take $\infty + 1 = \infty$ in DepthUpdate. Root i sets its depth to i as 0, and any other node updates its depth to i as $l + 1$ if it finds that the minimum depth to i buffered by its i-parents is l. Each node u periodically runs DepthUpdate so as to insure its buffered depths are close approximations to the real depths $L_i(u), i \in \mathcal{R}$. We will show that each node maintains just one incoming i-link over all time, so a node needs to contact just one i-parent when DepthUpdate is running.

We do not require that nodes update depths much more frequently than they sample targets. Assume depth updates by node u are triggered by three types of events:

- after u builds a new incoming i-link, immediately u runs DepthUpdate(u,i).
- after u samples a target, u runs DepthUpdate(u,j) for all $j \in \mathcal{R}$ immediately.
- after u is sampled as a target, u runs DepthUpdate(u,j) for all $j \in \mathcal{R}$ immediately.

Thus, nodes update depths about twice as frequently as their Poisson clocks tick.

4.2.2 Distributed greedy covering

In this section a greedy procedure is proposed to insure that each node has at least K incoming links with distinct colors. Notice that nodes are randomly sampling others. Assume a node u runs Procedure GreedyTreeCover after it samples node u_p as a target:

GreedyTreeCover does not use depth information. It has several properties: if Assumption 4.2.1 is valid, as nodes sample targets and run GreedyTreeCover,

1. statements in Assumption 4.2.1 remain valid;

Procedure GreedyTreeCover (Node u, Node u_p)

Output: return true if (V, E) changes, return false otherwise.
Tie broken: arbitrarily
If u has less than K incoming links and there exists i such that u has no incoming i-link but u_p has an incoming i-link,

- **Add:** if $d(u_p) < \bar{d}_{u_p}$, build (u_p, u) in E_i, and return true;
- **Insert:** if u_p has an i-child, say u_c, remove (u_p, u_c), build (u_p, u) and (u, u_c) in E_i, and return true.

Return false. (See Figure 4.3)

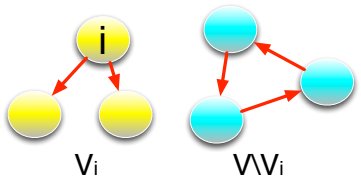


Figure 4.4: Cycles can appear at subgraph $(V \setminus V_i, E_i)$.

2. for each $i \in \mathcal{R}$, $|V_i|$ is nondecreasing, and the depth of tree (V_i, E_i) is also nondecreasing.

3. for each $i \in \mathcal{R}$ and each node u, both $d_i(u)$ and the number of incoming i-links of u are nondecreasing.

In the following we state several additional assumptions under which running GreedyTreeCover after nodes sample targets leads all nodes to be covered by K trees.

First, nodes have to provide enough fan-out degrees to meet the demands of incoming links, so Assumption 4.2.2 is assumed.

Assumption 4.2.2 (Minimum Degree) $\sum_{u \in V} \bar{d}_u \geq KN - M$.

Second, notice that for $i \in \mathcal{R}$, if at time 0 root i is not available and root i does not have outgoing i-links, it is not possible for any node to ever build incoming i-links from i. That is, $V_i = \{i\}$ over all time. To avoid that, we assume root i has at least one i-child at time 0:

Assumption 4.2.3 (Root Child Guarantee) *For each* $i \in \mathcal{R}$, $\bar{d}_i \geq 1$ *and at time* 0 *root* i *has at least one* i*-child.*

Third, GreedyTreeCover does not generate new cycles, but it cannot eliminate original cycles. As shown in Figure 4.4, $(V \setminus V_i, E_i)$ may contain cycles at time 0, which cannot be eliminated by GreedyTreeCover. In this section we avoid discussion of cycles by assuming Assumption 4.2.4 is valid; in the next section we will show how cycles are eliminated.

Assumption 4.2.4 (No Cycle) *At time 0 in (V, E), for any $i \in \mathcal{R}$, any node u with $d_i(u) = +\infty$ does not have incoming i-links.*

Assumption 4.2.4 implies that no cycle exists in $(V \setminus V_i, E_i)$ for each $i \in \mathcal{R}$ at time 0. By running GreedyTreeCover no i-links will be built between nodes in $V \setminus V_i$ and so no cycle ever appears. The following two indicate the convergence of running GreedyTreeCover.

Lemma 4.2.1 *If Assumptions 4.2.1 to 4.2.4 are all valid, GreedyTreeCover(u,v) returns false for all $u, v \in V$ if and only if $|\{i \in \mathcal{R} : u \in V_i\}| = K$ for each node u, that is, if and only if each node is covered by K trees.*

Proof. If a node has K incoming links with distinct labels, GreedyTreeCover returns false whichever target the node samples. So the if part follows.

Suppose there exists $u \in V, |\{i \in \mathcal{R} : u \in V_i\}| < K$. We prove the only if part by showing that GreedyTreeCover returns true when two specific nodes meet. Node u has fewer than K incoming links. Assumption 4.2.1 implies that each node has at most K incoming links. So $|E| \leq K(N-1)+(K-1)-M \leq \sum_{u \in V} \bar{d}_u - 1$ by Assumption 4.2.2. Thus there exists a node, say v, with $d(v) < \bar{d}_v$. One of the following two cases must hold: a) If v has K incoming links, GreedyTreeCover(u,v) returns true because "Add" can happen; b) If v has fewer than K incoming links, GreedyTreeCover(v,i) returns true if $v \notin V_i$ because "Insert" can happen. □

Proposition 4.2.1 *Under Assumptions 4.2.1 to 4.2.4, if GreedyTreeCover(u,v) runs whenever u samples v for any $u, v \in V$, then $|\{i \in \mathcal{R} : w \in V_i\}| = K$ for all $w \in V$ in finite time.*

Proof. Notice that statements in Assumptions 4.2.1 to 4.2.4 remain valid over all time. Whenever GreedyTreeCover returns true, $|E|$ increases by one. But $|E| \leq KN - M$. Proposition 4.2.1 follows from Lemma 4.2.1. □

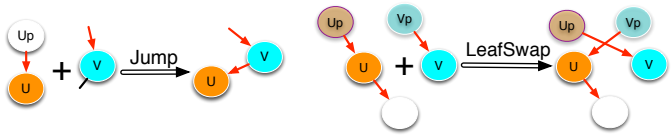


Figure 4.5: Single Tree Adjust.

GreedyTreeCover can achieve coverage, but it has two main drawbacks: first, the depth of the tree (V_i, E_i) for $i \in \mathcal{R}$ can be large; second, it cannot detect and eliminate cycles. In the next section we show how to solve these two problems by adding balance algorithms as complements.

4.2.3 Achieve balance inside trees

One way of decreasing the depth of a tree is to balance the tree. Here we provide a procedure called SingleTreeAdjust under which trees can achieve balance and cycles can be eliminated. Suppose SingleTreeAdjust(u,v) runs whenever node u samples node v as a target.

Procedure SingleTreeAdjust (Node u, Node v)

Output: return true if (V, E) changes, return false otherwise.
Tie broken: arbitrarily
If there exists i such that u, v both have incoming i-links,

- **Jump**: if $d(v) < \bar{d}_v$ and $l_i^u + 1 < l_i^v$, remove (u_p, u), build (v, u) in E_i, and return true;
- **LeafSwap**: if v is an i-leaf but u is not, and $l_i^u > l_i^v$, remove $(u_p, u), (v_p, v)$, build $(u_p, v), (v_p, u)$ in E_i, and return true.

Return false. (See Figure 4.5)

Procedure SingleTreeAdjust applies the information of buffered depths, which is updated periodically but may have estimation errors. Analyzing SingleTreeAdjust under depth updating is quite challenging. *To focus on properties of SingleTreeAdjust, let us temporarily assume Assumption 4.2.5 holds. Discussion under the case without Assumption 4.2.5 will be covered by simulation in Section 4.4.*

Assumption 4.2.5 (Instantaneous distance update) $\forall i \in \mathcal{R}, \forall u \in V$, *assume* $l_i^u = L_i(u)$ *over all time.*

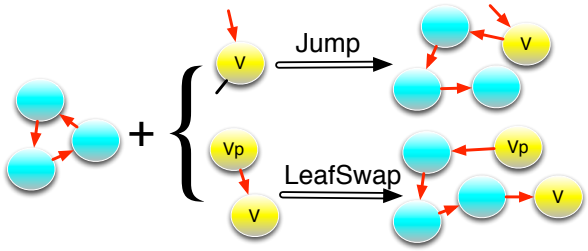


Figure 4.6: SingleTreeAdjust can eliminate original cycles.

For any nodes $u, v \in V$, after SingleTreeAdjust(u,v) runs:

1. Statements in Assumptions 4.2.1 to 4.2.4 remain valid if they are valid before running.

2. For any node, its number of incoming i-links and number of outgoing i-links both remain unchanged.

3. Under Assumption 4.2.5, for any node w with $d_i(w) \geq 1$, $L_i(w)$ is nonincreasing.

4. Under Assumption 4.2.5, no new cycles will be generated, and cycles in $(V \setminus V_i, E_i)$ are eliminated as shown in Figure 4.6, because $L_i(w) = +\infty$ for all $w \in V \setminus V_i$.

The following lemma shows that SingleTreeAdjust can return true unless all trees achieve balance.

Lemma 4.2.2 *Under Assumptions 4.2.1 and 4.2.5, if SingleTreeAdjust(u,v) returns false for all pairs of nodes u, v, for each $i \in \mathcal{R}$,*

1. *if Assumption 4.2.3 holds, Assumption 4.2.4 also holds,*

2. *for any two i-leaves u, v, $|L_i(u) - L_i(v)| \leq 1$, and*

3. *each i-internal node u is unavailable and $d_i(u) \geq 1$.*

Proof. 1) Together with Assumption 4.2.1, Assumption 4.2.3 implies that for each i there is at least one i-leaf. Thus Assumption 4.2.4 is valid; otherwise, SingleTreeAdjust(u,v) returns true if $u \in V \setminus V_i$ has an incoming i-link and v is an i-leaf.

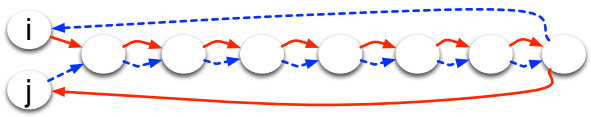


Figure 4.7: Mixed Nodes form a chain which may be very long. The two trees are each balanced but have large depth.

2) Assume nodes u and v_c are both i-leaves and $L_i(u)+1 < L_i(v_c)$. Assume v is the i-parent of v_c, then $L_i(u) < L_i(v)$. SingleTreeAdjust(u,v) returns true. Thus, the depths of any two i-leaves differ by at most one.

3) If u is an i-internal node, it cannot be an i-leaf because of 2), and it cannot be available otherwise SingleTreeAdjust(v,u) returns true where v is the i-leaf with the largest depth to i. □

Running GreedyTreeCover and SingleTreeAdjust together can achieve both coverage and balance, as well as eliminate cycles. However, balance in a tree does not guarantee the tree has small depth. As shown in Figure 4.7, if there are many nodes with a single outgoing i-link, chains may appear and thereby the depth of tree (V_i, E_i) can be large. Fortunately, there exist ways to eliminate conditions like that, as shown in the following.

4.2.4 Achieve balance among trees

Nodes with a single child play an important role in increasing the depth of a tree. An unavailable node u with a single i-child for certain $i \in \mathcal{R}$ either has $\bar{d}_u = 1$ or is a mixed node. The case where $\bar{d}_u = 1$ is less interesting because most nodes can be required to provide outdegree at least two, especially when the streaming rate of a substream is small compared to the upload capacity. Suppose Assumption 4.2.6 holds for simplicity.

Assumption 4.2.6 (Diversity Degree) $\bar{d}_u \neq 1$ *for all* $u \in V \setminus \mathcal{R}$.

Notice that in Assumption 4.2.6 existence of nodes with outdegree zero is allowed. Assumption 4.2.6 gets rid of the case that many nodes have outdegree one. Reducing the number of mixed nodes can lower the tree depths. In this section we provide Procedure MixedNodeAdjust under which the number of mixed nodes can be greatly reduced. For any pair of nodes u_c, v, suppose MixedNodeAdjust(u_c,v) runs when u_c samples v as a target.

Procedure MixedNodeAdjust (Node u_c, Node v)

Output: return true if (V, E) changes, return false otherwise.
Tie broken: arbitrarily.
If there exist $i, j, i \neq j$ such that v has a j-child say v_c, u_c has an i-parent say u, and

MixSwap: if $l_i^u \geq l_i^v, l_j^u \leq l_j^v$ and either of the two is true:

a) $l_i^u \neq l_v^i$ or $l_j^u \neq l_j^v$,

b) $l_i^u = l_i^v$, $l_j^u = l_j^v$, $(u - v)(j - i) > 0$, (Note u, v, i, j are ids in {1,...N}.)

then remove $(u, u_c), (v, v_c)$, build (u, v_c) in E_j, build (v, u_c) in E_i, and return true.
Otherwise return false. (See Figure 4.8)

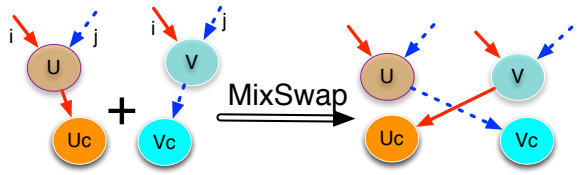


Figure 4.8: u_c and v_c switch their parents if one can decrease its depth while the other one's depth does not increase, or depths are unchanged but lower-id parents can get lower-color links.

MixedNodeAdjust(u_c,v) returns true if either u_c or some child v_c of v gets a strictly smaller depth in either tree i or tree j, or the depths of u_c and the children of v remain the same but the parent u or v with lower id gains an outgoing link of the lower color, i or j. We break the tie by assuming parents with lower ids have a preference on links with lower colors, so as to eliminate the case that there are many mixed nodes with exactly the same depths in multiple trees.

Under Assumption 4.2.5, for any pair of nodes u, v, after MixedNodeAdjust(u,v) runs,

1. Statements in Assumptions 4.2.1 to 4.2.4 remain valid if they are valid before running.

2. $L_i(w)$ is nonincreasing for any node w.

3. No new cycle is generated.

Moreover, the following lemma indicates that MixedNodeAdjust can return true unless the depth vectors of mixed nodes for any pair of trees i, j form a

strict chain:

Lemma 4.2.3 *Under Assumption 4.2.5, if MixedNodeAdjust($\bar{u}$,$\bar{v}$) returns false for all pairs of nodes $\bar{u}, \bar{v} \in V$, for any $i, j \in \mathcal{R}$ and any two mixed-i-j-nodes u, v, either $(L_i(u), L_j(u)) < (L_i(v), L_j(v))$ or $(L_i(v), L_j(v)) < (L_i(u), L_j(u))$.*

Proof. The lemma follows by noticing that if u, v are both mixed-i-j-nodes, "MixSwap" can happen when either a child of u contacts v or a child of v contacts u, unless $(L_i(u), L_j(u))$ and $(L_i(v), L_j(v))$ are in a strict order. □

Lemma 4.2.3 still does not guarantee that the number of mixed nodes is small. As shown in Figure 4.7, mixed nodes can form a long chain where the conclusion in Lemma 4.2.3 still holds. However, the appearances of the structure in Figure 4.7 are quite rare because random sampling is assumed. And intuitively they become rarer as N increases. In practice we may safely ignore the anomaly, but here for completeness of analysis, we eliminate the possibility of a long chain as in Figure 4.7 by assuming Assumption 4.2.7 holds. We will show later by simulation that ignoring Assumption 4.2.7 does not harm performance.

Assumption 4.2.7 (Shower head) *At time 0, there exists $c \in \mathcal{Z}^+$ such that for each $i \in \mathcal{R}$, there are at least M i-leaves in $(\hat{V}_i, E_i)$, where $\hat{V}_i := \{u \in V : d_i(u) \leq c\}$.*

Assumption 4.2.7 says that initially in any tree the subtree of nodes with depth bounded by c has at least M leaves. Intuitively Assumption 4.2.7 suggests there is something analogous to a shower head, which provides enough branches near the top of each tree (V_i, E_i). The value c in Assumption 4.2.7 can be as small as $O(\log M)$, or even $O(1)$ if the root or its children have large outdegrees. We have Lemma 4.2.4 under Assumption 4.2.7.

Lemma 4.2.4 *Under Assumptions 4.2.1 and 4.2.5 to 4.2.7, if SingleTreeAdjust(u,v) and MixedNodeAdjust(u,v) both return false for all $u, v \in V$, then for each $i \in \mathcal{R}$ the depth of each tree (V_i, E_i) is less than or equal to $\log_2(N+1) + c$, where c is defined in Assumption 4.2.7.*

Proof. Suppose the depth of tree (V_i, E_i) is $\bar{l}_i$. Lemma 4.2.3 implies that for each $j \neq i$ and for each $k \leq \bar{l}_i$, there is at most one mixed-i-j-node whose

depth to i is k. So there are at most $M-1$ mixed nodes which have exactly one i-child and whose depth to i is k. Lemma 4.2.2 and Assumption 4.2.6 imply that all i-internal nodes must have at least two children and at least one i-child. Thus, for each $k \leq \bar{l}_i - 2$, in the set of nodes whose depth to i is k, 1) each of them has at least one i-child and 2) at most $M-1$ of them can have a single i-child, while each of the other nodes has at least 2 i-children because they are unavailable non-mixed i-internal nodes.

Notice that there are at least M leaves, say nodes $w_1, w_2, ...w_M$, in $(\hat{V}_i, E_i)$, where $\hat{V}_i$ is defined in Assumption 4.2.7. Let G_k be the subtree rooted at w_k of (V_i, E_i). Then subtrees $G_1, G_2, ...G_M$ are mutually disjoint. Consider two cases: 1) Assume $c \leq \bar{l}_i - 2$. Each of $G_1, ...G_M$ has at least one node with depth to i being c; otherwise there is one i-internal node without i-child in one subtree. So there are at least M nodes whose depth to i is c, among which at least one has two i-children. The number of nodes whose depth to i is in $[c, \bar{l}_i - 1]$ is at least $1+2+4+...+2^{\bar{l}_i-1-c} \leq N$, so $\bar{l}_i \leq \log_2(N+1)+c$; 2) Assume $c \geq \bar{l}_i - 1$. Then $\bar{l}_i \leq \log_2(N+1)+c$ also holds since $N \geq 1$. $\square$

4.2.5 Combining everything

Here we combine all parts above together. For each pair of nodes u, v, run CombinedUpdate(u,v) when u samples v.

Procedure CombinedUpdate (Node u, Node v)

Output: return true if (V, E) changes, return false otherwise.
Tie broken: arbitrarily

Nodes u, v update their buffered depth by running DepthUpdate for each $i \in \mathcal{R}$ respectively
return GreedyTreeCover(u, v) **or**
SingleTreeAdjust(u, v) **or** MixedNodeAdjust(u, v)

Just like that in C or C++, in CombinedUpdate, if operation "A" returns true, operation "A or B" immediately returns and operation "B" does not run.

Notice that buffered depths are also updated whenever new links are built, as assumed in Section 4.2.1. And notice that under Assumption 4.2.5, for any pair of nodes u, v, after CombinedUpdate(u,v) runs, statements in Assump-

tions 4.2.1 to 4.2.4, 4.2.6 and 4.2.7 remain valid if they hold before running, respectively.

Lemma 4.2.5 shows that by running CombinedUpdate a certain metric changes monotonically. Some additional definitions are necessary before defining the metric. Define $L_i'(u) = \min\{L_i(u), N\}$, i.e., $L_i'(u)$ is the same as $L_i(u)$ except that $L_i'(u) = N$ instead of $+\infty$ if there is no path from i to u. Let $Y = \sum_{u \in V} \sum_{i \in \mathcal{R}} L_i'(u)$ so Y is the sum of all modified depths of all nodes. Let $D(u) := \sum_{i \in \mathcal{R}} i \cdot d_i(u)$ be the sum of the colors of all outgoing links of u, and let $S := \sum_u uD(u)$.

Lemma 4.2.5 *Under Assumption 4.2.5, for any pair of nodes u, v, after CombinedUpdate(u,v) runs, if it returns true, $(-|E|, Y, -S)$ decrease lexicographically by at least one, otherwise $(-|E|, Y, -S)$ does not change.*

Proof. Under Assumption 4.2.5, if GreedyTreeCover returns true, $|E|$ increases by one; if SingleTreeAdjust returns true, or MixedNodeAdjust returns true because of Condition (a), Y decreases by at least one but $|E|$ remains unchanged; if MixedNodeAdjust returns true because of Condition (b), S increases by at least one while $|E|, Y$ remain unchanged. □

Lemma 4.2.5 helps to show convergence of the algorithm.

Proposition 4.2.2 *Under Assumptions 4.2.1 to 4.2.3 and 4.2.5 to 4.2.7, suppose CombinedUpdate(u,v) runs whenever u samples v for any $u, v \in V$. Then in finite time CombinedUpdate(u,v) returns false for all u, v, and at that time,*

(a) $\forall w \in V, |\{i \in \mathcal{R} : w \in V_i\}| = K$,

(b) $\forall i \in \mathcal{R}$, *the depth of the tree (V_i, E_i) is bounded by* $\log_2(N+1) + c$, *where c is the value in Assumption 4.2.7.*

Proof. Notice that statements in Assumptions 4.2.1 to 4.2.3, 4.2.6 and 4.2.7 are valid over all time. Lemma 4.2.5 and the boundedness of $|E|, Y$ and S imply that in finite time CombinedUpdate will return false whenever any two nodes meet. Lemma 4.2.1 implies that (a) is valid while Lemma 4.2.4 implies that (b) is valid. □

4.2.6 Comment on distributed depth update

Under Assumption 4.2.5, no new cycles can appear by running CombinedUpdate, which is not the case when depths are updated distributively. Here we argue that cycles are rare and can be eliminated quickly even if Assumption 4.2.5 does not hold.

First, a new cycle appears only if a node builds an incoming link from one of its descendants, which happens only if the descendant has a smaller buffered depth. That condition is rare because 1) most nodes just have a few descendants; 2) if u is a descendant of v and if the depth of u is larger than the depth of v, by running SingleTreeAdjust and MixedNodeAdjust the depth of u can remain larger than that of v. Usually a node has smaller depth than its ancestors only when many nodes suddenly become ancestors of the node in a short time, which is also a rare event.

Second, even if a new cycle appears, it will disappear in a short time. By DepthUpdate, depths of nodes in a cycle keep updating and can count to a large value, just like the "counting to infinity" problem in network routing. Whenever a node with a large depth meets a leaf node, changes as shown in Figure 4.6 can happen and thereby the cycle disappears. At least half of the nodes in a tree are leaves, so by random sampling, cycles are eliminated quickly.

In summary, we argue that distributed depth update does not harm performance much compared to that under Assumption 4.2.5, which is supported by simulations in Section 4.4.

4.3 Rate of convergence analysis

Analyzing the complexity for message exchange for the algorithm is quite challenging. This section provides a stochastic bound for the convergence time under the case of a single tree, i.e., $M = K = 1$. We further assume that each node has outdegree at least 2, i.e., $\bar{d}_u \geq 2$ for all $u \in V$. And suppose Assumption 4.2.5 holds so that each node knows its depth to the root. For theoretical tractability, instead of running CombinedUpdate, we simplify the algorithm by assuming that only "Add" in GreedyTreeCover and "Jump" in SingleTreeAdjust run when two nodes meet.

The simplified model is summarized as follows: each node knows its depth

to the root; whenever a node's Poisson clock ticks, the node samples a target uniformly at random; if the target is available and the depth of the target is less than the depth of the node by at least two, the node removes its current incoming link if there is one and builds a new incoming link from the target; otherwise, nothing happens. Assume initially each node has at most one incoming link, then convergence time is upper bounded:

Proposition 4.3.1 *Let T be the first time for the maximum depth of all nodes to be less than or equal to $\lceil \log_2(N+1) \rceil$, then for any $\epsilon > 0$,*

$$P\left[T > 21 \log_2(N+1) + 16\epsilon\right] < 3e^{-\epsilon}.$$

The maximum depth is $+\infty$ if there is a node to which no path exists from the root, so Proposition 4.3.1 bounds the time for the tree to cover all nodes and achieve balance. It implies convergence in $O(\log N)$ time.

We argue that for the case of multiple trees similar bounds as in Proposition 4.3.1 can be generated, and by running CombinedUpdate the network can converge in $O(\log N)$ time. Because when targets are unavailable, "LeafSwap" substitutes "Jump" efficiently since half of the nodes are leaves, and "Insert" works well when "Add" does not work.

The proof of Proposition 4.3.1 is provided below.

4.3.1 Proof of Proposition 4.3.1

We assume nodes sample targets randomly at times of Poisson processes with rate 1. Equivalently, we can assume that each $\langle node, node \rangle$ pair maintains a Poisson clock which ticks at rate $\frac{1}{N}$. The following definitions are applied for the proof.

Definition 4.3.1 *Define $l_f := \lfloor \log_2(N+1) \rfloor - 1$ and $l_c := \lceil \log_2(N+1) \rceil - 1$.*

Define $l_\alpha := \lceil \log_2((1-\alpha)N + 1) \rceil - 1$.

Define $Z_i, i \geq 0$ to be the number of nodes with depths $\leq i$. Note that $Z_i(t)$ is a counting process.

Define $Z_{-1} = \tilde{Z}_{-1} \equiv 0$.

Denote by $Poi(\lambda)$ a Poisson random variable with mean λ.

Denote $\max\{A, 0\}$ for expression A by $(A)^+$.

Lemma 4.3.1 describes a motion trajectory to be applied in the proof.

Lemma 4.3.1 *Let* $\mathbf{y}(t) = (y_0(t), y_1(t), ...y_k(t))$*, with parameters* $k \in \mathbb{Z}^+$ *and* $\beta \in \mathbb{R}^+$*, be the motion trajectory characterized by:*

$$y_i'(t) = \beta \left(1 + 2y_{i-1}(t) - y_i(t)\right)^+, s.t$$
$$y_{-1}(t) \equiv 0, y_0(0) = 1, y_i(0) \geq 0, 1 \leq i \leq k,$$

then

$$y_i(t) \geq 2^{i+1} \left\{1 - P[Poi(\beta t) \leq i - 1]\right\} - 1.$$

Proof. Let $\delta_i(t) := \left[2^{i+1} - 1 - y_i\left(t/\beta\right)\right]/2^{i+1}$, we can simplify the equation of y as

$$\delta_i'(t) = \left(\delta_{i-1}(t) - \delta_i(t)\right)^-.$$

Notice that by induction on i we can show that $\delta_i(t) \leq \theta_i(t)$, where

$$\theta_i(t) = \theta_{i-1}(t) - \theta_i(t), \theta_i(0) = \delta_i(0).$$

Solving the differential equation about θ gives

$$\delta_i(t) \leq \theta_i(t) = e^{-t} \sum_{k=0}^{i-1} \theta_{i-k}(0) \frac{t^k}{k!}.$$

$\forall i, \theta_i(0) = \frac{2^{i+1} - 1 - y_i(0)}{2^{i+1}} \leq 1$. Thus

$$2^{i+1}\delta_i(t) \quad \leq \quad 2^{i+1}\theta_i(t) = 2^{i+1} P[Poi(t) \leq i - 1],$$

and so the lemma follows. □

The model describes a Markov process with state being $G = (V, E)$. We apply $G = (V, E)$ to denote the process as well as the graph. It is not difficult to see that graph $G = (V, E)$ can converge to a balanced tree covering all nodes in finite time, because 1) the depth of each node is nonincreasing, so that Z_i for each i is nondecreasing, and 2) there exist nodes which can decrease their depths if (V, E) is not balanced or (V, E) does not cover all nodes.

The process is separated into two phases, addressed by Lemmas 4.3.2 and 4.3.3, respectively. The time for the first phase is described in Lemma 4.3.2.

Lemma 4.3.2 *For any $\alpha \in (0,1)$, let T_0 be the first time $Z_{l_\alpha} \geq (1-\alpha)N$. Then for any time $t \geq 0$,*

$$P[T_0 > t] \leq 2^{l_\alpha+1} P[Poi(\alpha t/2) \leq l_\alpha - 1].$$

Proof. Define an alternative process $G' = (V', E')$ such that it is identical to the original process G when $t < T_0$; when $t \geq T_0$, whenever a node with depth greater than l_α changes its depth, a new node with fan-out degree 2 whose depth is ∞ arrives to G'. After T_0, $|V'|$ may increase but we assume the Poisson clock of each $(node, node)$ pair still ticks at rate $\mu = 1/N$. On G', the number of nodes with depths greater than l_α does not change after T_0, and is always larger than or equal to αN. The probability $P[T_0 > t]$ is identical for processes G or G'. *In the following, we discuss process G'. For simplicity, we apply the same notations for G' as for G.* Let $Z = (Z_0(t), Z_1(t), ...Z_{l_\alpha}(t))$.

Notice that the number of available nodes with depths less than or equal to $i-1$ is greater than or equal to $(1 + 2Z_{i-1} - Z_i)^+/2$: consider each node labels its outgoing degrees and marks the first two degrees red. Each node with depth less than or equal to $i-1$ has at least 2 red degrees, but there are only $Z_i - 1$ nodes to serve. So there are at most $(Z_i - 1)/2$ nodes whose red degrees are both taken.

The number of nodes with depths greater than l_α is greater than or equal to αN. Thus, if $i \leq l_\alpha$, the transition rate for Z_i to jump is lower bounded by

$$\mu\alpha N \frac{(1 + 2Z_{i-1} - Z_i)^+}{2N} = \frac{\alpha}{2}(1 + 2Z_{i-1} - Z_i)^+.$$

There exists a process $\tilde{Z} = (\tilde{Z}_0(t), \tilde{Z}_1(t), ...\tilde{Z}_{l_\alpha}(t))$ in $\mathcal{Z}_+^{l_\alpha+1}$ on an extended probability space such that each coordinate of $\tilde{Z}$ has jumps of size one and jump rate for $\tilde{Z}_i$ is $\alpha(1 + 2\tilde{Z}_{i-1} - \tilde{Z}_i)^+/2$. Notice that simultaneous jumps of different coordinates of $\tilde{Z}$ are allowed. Let $\tilde{Z}(0) = (1, 1, 1, ...1)$. Initially we have $Z(0) \geq \tilde{Z}(0)$.

Process Z and $\tilde{Z}$ can be coupled so that $Z(t) \geq \tilde{Z}(t)$ with probability one for all t. That is because if $Z \geq \tilde{Z}$ then the jump rate of Z_i is greater than or equal to the jump rate of $\tilde{Z}_i$ for all i such that $Z_i = \tilde{Z}_i$. So the jumps of

$\tilde{Z}$ can be obtained by generally thinning the jumps of Z, and adding more jumps to $\tilde{Z}_i$'s with $Z_i > \tilde{Z}_i$.

By induction on i it is easy to show that $\tilde{Z}_i(t) \leq 2^{i+1} - 1$ with probability one for all t, because the jump rate of $\tilde{Z}_i$ is zero if $\tilde{Z}_i = 2^{i+1} - 1$. Moreover, $\forall i \in [0, l_\alpha]$,

$$\begin{aligned} \frac{dE[\tilde{Z}_i(t)]}{dt} &= E\left[\frac{\alpha}{2}\left(1 + 2\tilde{Z}_{i-1}(t) - \tilde{Z}_i(t)\right)^+\right] \\ &\geq \frac{\alpha}{2}\left(1 + 2E[\tilde{Z}_{i-1}(t)] - E[\tilde{Z}_i(t)]\right)^+. \end{aligned}$$

Let $\mathbf{y}(t) = (y_0(t), y_1(t), ...y_{l_\alpha}(t))$ be the motion trajectory defined in Lemma 4.3.1, with parameters $\beta = \alpha/2$, $k = l_\alpha$, and boundary conditions $y_i(0) = 1$ for all $0 \leq i \leq l_\alpha$. By induction on i it is easy to prove that $E[\tilde{Z}_i(t)] \geq y_i(t)$.

Define $\Delta(t) := \left(2^{l_\alpha+1} - 1 - \tilde{Z}_{l_\alpha}(t)\right)$. Notice that $\Delta(t) \geq 0$, apply Markov's inequality and Lemma 4.3.1,

$$\begin{aligned} & P[T_0 > t] = P[Z_{l_\alpha}(t) \leq (1-\alpha)N - 1] \\ \leq\ & P[\tilde{Z}_{l_\alpha}(t) \leq 2^{l_\alpha+1} - 2] = P[\Delta(t) \geq 1] \\ \leq\ & E[\Delta(t)] \leq 2^{l_\alpha+1} - 1 - y_{l_\alpha}(t) \\ \leq\ & 2^{l_\alpha+1} P[Poi(\alpha t/2) \leq l_\alpha - 1]. \end{aligned}$$

Lemma 4.3.2 follows. □

The time of the second phase is described by Lemma 4.3.3.

Lemma 4.3.3 *For any $\alpha \in (0, 0.5)$, given $Z_{l_\alpha}(0) \geq (1-\alpha)N$, let T_1 be the first time that $Z_{l_c+1} = N$, then*

$$P[T_1 \leq t] \geq \left[1 - e^{-(1-2\alpha)t/2}\right]^{\alpha N}.$$

Proof. Let X_i be the number of nodes with depths equal to i, and let Y_i be the number of available nodes with depths less than or equal to i.

Consider the jumping rate of Z_{l_c+1}. Notice that $Z_{l_\alpha}(t) \geq (1-\alpha)N$, and

$$\begin{aligned} & 2Y_{l_c} \geq \max_{i \in [l_\alpha, l_c]} \{1 + Z_i - X_{i+1}\} \\ \geq\ & \max_{i \in [l_\alpha, l_c]} \left\{(1-\alpha)N + \sum_{k=l_\alpha+1}^{i} X_k - X_{i+1}\right\} \geq (1-2\alpha)N. \end{aligned}$$

The last inequality above is due to the fact that $\sum_{k=l_\alpha+1}^{l_c+1} X_k \leq \alpha N$.

The rate for any node with depth greater than $l_c + 1$ to jump to join Z_{l_c+1} is at least $\mu Y_{l_c} \geq (1 - 2\alpha)/2$. So the lemma follows. □

Now Lemmas 4.3.2 and 4.3.3 are combined to prove Proposition 4.3.1. Consider Lemma 4.3.2, and apply the Chernoff bound for Poisson variables: $P[Poi(\lambda) \leq x] \leq \frac{e^{-\lambda}(\lambda e)^x}{x^x}$ if $\lambda > x$, and $l_\alpha - 1 < l_\alpha \leq \log_2(N+1)$, we have

$$P[T_0 > 2t/\alpha] \leq 2\exp\left\{r(1 + ln2 - \frac{t}{r} + ln\frac{t}{r})\right\},$$

where $r = \log_2(N+1)$. Notice that $1 + ln2 - \frac{t}{r} + ln\frac{t}{r} \leq -0.2 - 2(t/r - 3)/3 = -2(t/r - 2.8)/3$, so

$$P\left[T_0 > \frac{2}{\alpha}\left(2.8\log_2(N+1) + \frac{3}{2}\epsilon\right)\right] \leq 2e^{-\epsilon}.$$

Consider Lemma 4.3.3:

$$P[T_1 \leq t] \geq 1 - \alpha N e^{-(1-2\alpha)t/2} \geq 1 - (N+1)e^{-(1-2\alpha)t/2}.$$

And so

$$P\left[T_1 > \frac{2\big(ln(N+1) + \epsilon\big)}{1 - 2\alpha}\right] \leq e^{-\epsilon}.$$

Choose $\alpha = 0.36987$, which minimizes $5.6/\alpha + 2\ln 2/(1 - 2\alpha)$, and we get

$$P\left[T > 21\log_2(N+1) + 16\epsilon\right] \leq 3e^{-\epsilon}.$$

4.4 Validating the algorithm by simulation

We show that CombinedUpdate works pretty well under Assumptions 4.2.1 to 4.2.3 and 4.2.6, without Assumptions 4.2.4, 4.2.5 and 4.2.7. Let each node sample target randomly with rate $\mu = 1$ and run CombinedUpdate. Assumption 4.2.5 is not invoked so depths update distributively. Notice that CombinedUpdate runs instantaneously in simulations. In each experiment below, we set $N = 1000$ fixed; at time 0, we first set E to be empty, then

let each root i build an i-child which is randomly selected from $V \setminus \mathcal{R}$. So at time 0 E contains $|\mathcal{R}|$ links and Assumptions 4.2.1 and 4.2.3 both hold. We let Assumptions 4.2.2 and 4.2.6 hold too.

Because Assumption 4.2.1 holds, during each simulation below (V_i, E_i) for each i is always a tree. Tree i is given by (V_i, E_i). Say that a node is *covered by tree* i if $L_i(u) < +\infty$, and say that a node is *fully covered* if it is covered by at least K trees.

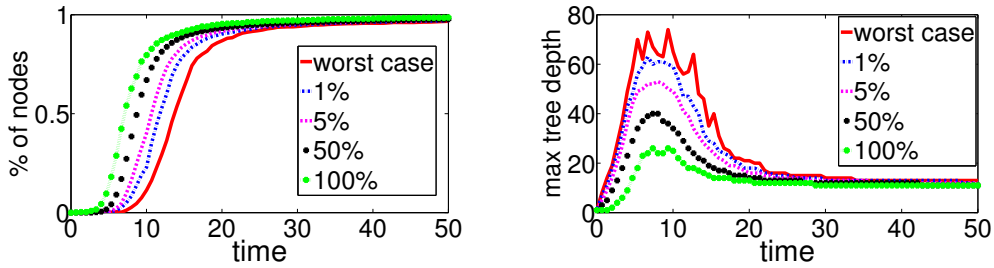
In experiments below, the parameters chosen include K, M and the degree vector $(\bar{d}_u)_{u\in V}$. For each selection of parameters $K, M, (\bar{d}_u)_{u\in V}$, we repeat running the experiment 500 times, with each experiment running for 100 time units and with system states recorded in the same time units. Metrics considered include the fraction of nodes fully covered and the maximum tree depth.

4.4.1 Homogeneous degrees and tight capacity constraint

In this series of experiments, we let each root have degree $K-1$: $\forall i \in \mathcal{R}, \bar{d}_i = K-1$, and let each non-root node have degree K: $\forall u \in V \setminus \mathcal{R}$, $\bar{d}_u = K$. The capacity is tight because the equality in Assumption 4.2.2 is achieved: $\sum_{u\in V} \bar{d}_u = KN - M$. Keep $K \geq 2$ so Assumption 4.2.6 holds. After repeating an experiment for 500 times, for $a = 0.2, 1, 5, 50, 100$, we record metrics of the $a\%$ worst experiments at each time t. Notice that each line with legend "worst case" corresponds to the case $a = 0.2$, which means that there is no experiment performing worse than the line at any time. $(0.2\% * 500 = 1)$.

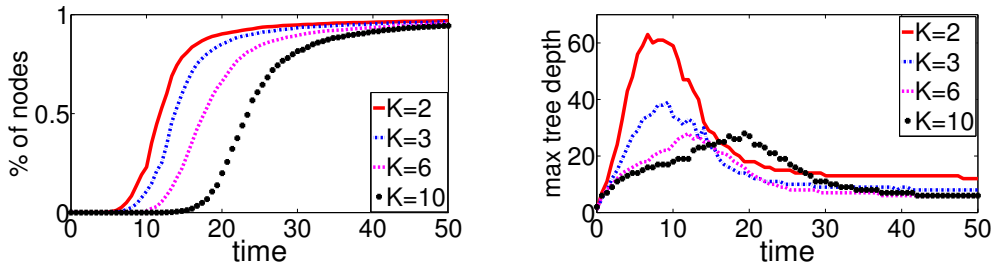
For example, in Figure 4.9 we set $M = K = 2$. A point (t, y) in Figure 4.9(a) on the line with legend "5%" means that in 5% of 500 repeated experiments, the fraction of nodes fully covered at time t is no larger than y; a point (t, y) in Figure 4.9(b) on the line with legend "1%" means that in 1% of 500 repeated experiments, the max tree depth at time t is no less than y.

Figure 4.9 shows what a specific sample path looks like under CombinedUpdate. In Figure 4.9(a), we can see that the fraction of nodes fully covered increases almost exponentially from 0 to 1, over 90% nodes are fully covered by time 25 under 99% experiments. That is because nodes can gradually



(a) % of nodes fully covered (b) max tree depth

Figure 4.9: Cumulative distribution when $M = K = 2$. Point (t, y) on a line legended $a\%$ means the corresponding metric of time t is worse than y in only $a\%$ of 500 experiments.



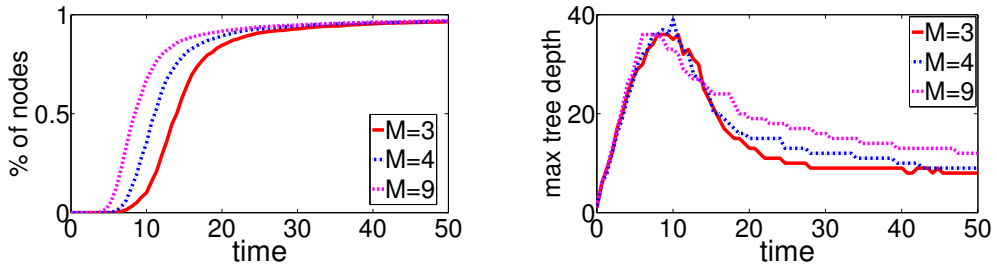
(a) % of nodes fully covered (b) max tree depth

Figure 4.10: 1% cumulative lines when $M = K$ varies.

increase the number of trees covering them until they get fully covered, as they meet other fully covered nodes. *It appears that the fraction of nodes is almost nondecreasing over all time, which validates that cycles generated are rare and are quickly eliminated.* As indicated in Figure 4.9(b), the maximum tree depth increases linearly in the beginning, then decreases almost exponentially, and finally converges to below 12. At time 25, in 99% repeated experiments max tree depths are below 20. The rate of convergence follows Proposition 4.3.1, though Proposition 4.3.1 is for the case of one tree.

Notice that not only in Figure 4.9, but also in all our simulations below, the "worst case" lines are quite close to the "1%" lines, but the latter are much more smooth than the former. *In the following, we apply "1%" lines instead of "worst case" lines to describe performance.*

In Figure 4.10 we test different K's so as to make sure convergence follows in other cases. Notice that typically in practice K is below 10. Set $M = K$ with K varying in $\{2, 3, 6, 10\}$. We expect to observe similar images as in



(a) % of nodes fully covered

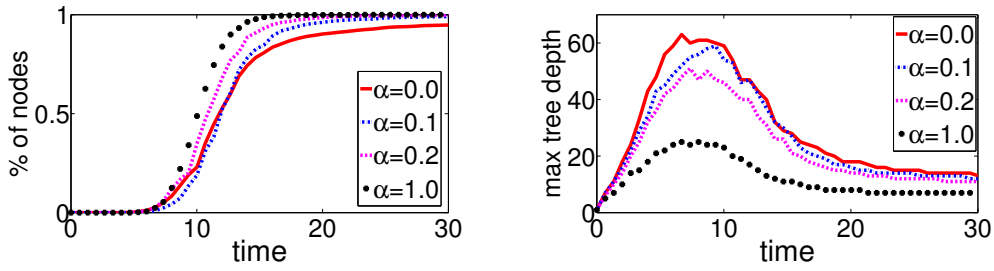
(b) max tree depth

Figure 4.11: 1% cumulative lines when M varies and $K = 3$.

Figure 4.9, with longer convergence time as K increases because each node has to get covered by more trees. For each K, we draw the 1% worst case line in Figure 4.10. As expected, for each K both the fraction of nodes fully covered and the max tree depth converge, as in Figure 4.9. In Figure 4.10(a), lines shift right almost linearly with a slow rate as K increases. With $K = 10$, over 90% nodes are fully covered by time 40. In Figure 4.10(b), lines shift both downwards and right, and converge to lower values as K increases. Because balanced trees have smaller depth if nodes have larger degree: with $K = 10$, the line converges to 4 in Figure 4.10(b). Whatever K is, the max tree depth is below 20 by time 25.

In Figure 4.11, we test the case under source coding by drawing the 1% worst case lines when $K = 3$ and M is in $\{3, 4, 9\}$, that is, there are more trees than nodes need. Notice that the capacity is still tight. In Figure 4.11(a), lines shift left as M increases, showing that source coding tends to decrease the convergence time. In Figure 4.11(b), limits slightly increase as M increases, because when M is larger there are more types of mixed nodes, which may have single children in a tree and thereby increase the depth. That condition is also implied by Proposition 4.2.2, where the value c is considered to be of $O(\log M)$. Thus, source coding creates a tradeoff between the tree depth limit and the convergence time of coverage. Intuitively and as shown in Figure 4.11(b), the increasing of depth limit is of $O(\log M)$ which is small, so it is worth trying source coding to get a faster convergence.

For all simulations above, we test cases where the capacity is tight, which illustrates Proposition 4.2.2 and supports that convergence is exponential. One common feature of curves in Figures 4.9(a), 4.10(a) and 4.11(a) is that long tails exist. For example, in Figure 4.11(a), it takes quite long for curves



(a) % of nodes fully covered

(b) max tree depth

Figure 4.12: 1% cumulative, $M = K = 2$ and α varies.

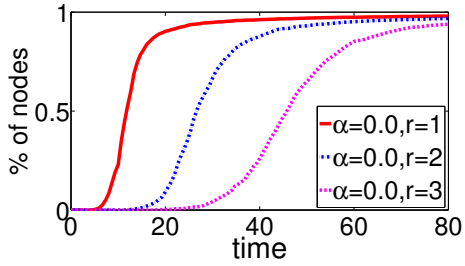
to arrive at 1. That is because near the end of the process only a few nodes are available and a few others are not fully covered, and it takes long for these nodes to meet each other by random sampling. Long tails can be eliminated by broadcasting or adding more capacity. Broadcasting is not discussed; in the following we show experiments where extra capacity exists.

4.4.2 Loose capacity constraint eliminates the long tail

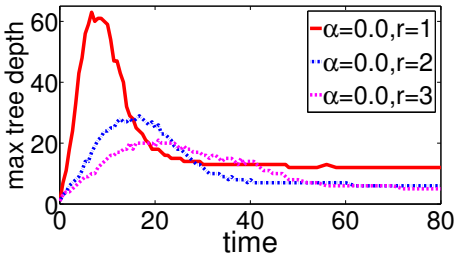
We set the total number of degrees $\sum_{u \in V} \bar{d}_u = (1 + \alpha)NK$, with a new parameter α. To achieve that, we first let each node have degree K, then add αNK degrees, one by one, to nodes selected uniformly at random. In Figure 4.12, we set $M = K = 2$, and let α increase. In Figure 4.12(a), we can see that adding just 10% extra capacity can greatly shorten the tail: all nodes are fully covered by time 25 as shown by the line $\alpha = 0.1$. The larger α is, the shorter the tail is. When $\alpha = 1.0$, all nodes get fully covered by time 15. In Figure 4.12(b), as α increases, curves converge faster and limits also decrease.

4.4.3 Polarized degrees for the server-client case

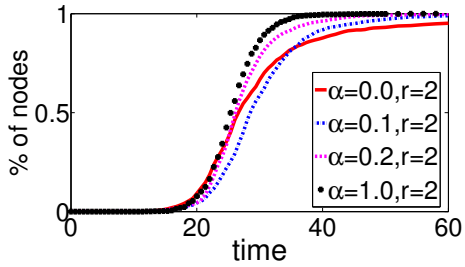
In this section, we show that CombinedUpdate works well under the server-client case, where a portion of nodes are server nodes with high degrees and other nodes are client nodes with zero degrees. The algorithm favors nodes with higher degree; they tend to get smaller depths than nodes with lower degrees. We expect to observe similar images as in Figure 4.9. Notice that the convergence times will increase because it takes more time for nodes to meet



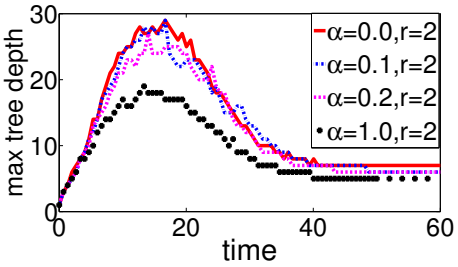
(a) % of nodes fully covered



(b) max tree depth

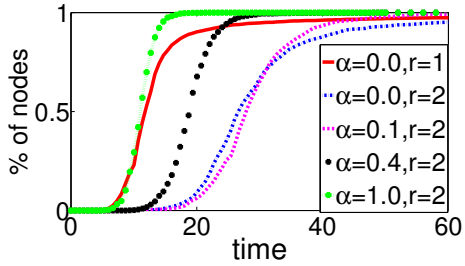


(c) % of nodes fully covered

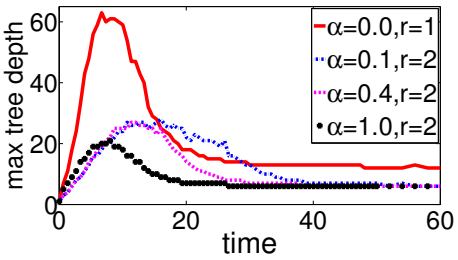


(d) max tree depth

Figure 4.13: 1% cumulative. $M = K = 2$, N/r server nodes with degree rK, αNK degrees added to server nodes randomly.



(a) % of nodes fully covered



(b) max tree depth

Figure 4.14: 1% cumulative. $M = K = 2$, $\frac{1+\alpha}{r}N$ server nodes with degree rK.

server nodes under uniform random sampling. That problem can be solved by adding mechanisms to help nodes find server nodes. In this thesis, we stay focused on the uniform sampling assumption despite the small increase in convergence time.

In Figure 4.13, we let N/r server nodes (including roots) have degree rK and all other nodes have degree 0, then add αNK degrees one by one to server nodes randomly. As expected, in Figure 4.13(a), lines shift right as r increases because there are fewer server nodes, but still increase exponentially from 0 to 1. In Figure 4.13(b), max tree depth decreases greatly as r increases from 1 to 2, and further decreases as r increases. Figures 4.13(c) and 4.13(d), in which r is fixed at 2 with increasing α, show that long tails like those in Figure 4.12 are eliminated.

In Figure 4.14, we let $\frac{1+\alpha}{r}N$ server nodes (include roots) have degree rK and all other nodes have degree 0. We set $r = 2$, let α increase, and draw the line at $\alpha = 0, r = 1$ for comparison. Notice that lines at $\alpha = 0$ in Figures 4.13 and 4.14 are exactly the same. As α increases, there are more server nodes so lines shift left quickly in Figure 4.14(a), and max tree depth decreases quickly in Figure 4.14(b). When $\alpha = 0.1, r = 2$, lines in Figure 4.14 are close to lines in Figure 4.13. As α increases, in Figure 4.14 lines shift left but in Figure 4.13 they do not. That suggests that performance is sensitive to the number of server nodes instead of the degree distribution among server nodes.

Above all, our simulations validate Propositions 4.2.2 and 4.3.1 by showing that the fraction of nodes fully covered increases from 0 to 1, and the maximum tree depth decreases to its limit, almost exponentially, under tight or un-tight capacity constraint, homogeneous or heterogeneous capacity distribution, when Assumptions 4.2.1 to 4.2.3 and 4.2.6 hold. The simulations suggest that cycles are eliminated quickly because the fraction of nodes covered is almost increasing over all time; long tails can be eliminated by adding 10% extra capacity; and the algorithm favors nodes with higher degree so it works well under the server-client case too. Convergence times increase with either increasing K or decreasing the chance for nodes to meet server nodes. When parameters change, curves shift with shapes staying similar, revealing robustness of the algorithm.

Chapter 5

CONCLUSIONS AND FUTURE WORK

In this thesis, we discuss stability regions of BitTorrent-like P2P networks under different methods, i.e., providing pieces to peers upon arrival, requiring peer seeds to dwell, using network coding, using arbitrary work-conserving piece selection policies, bundling files as universal swarms, etc. The assumption of random peer sampling is applied on all stochastic models. It is shown that stability regions can be generated based on the missing piece syndrome, which turns out to be the effect that limits stability.

Besides discussions on scalability of file sharing networks, scheduling algorithms for P2P streaming are also covered, in a flow level model assuming random peer sampling, as for file sharing. A distributed algorithm for multicast tree management is presented, to achieve coverage and balance within delay logarithmic in the number of peers.

Further works on P2P file sharing and streaming networks can extend the work of this thesis. Several interesting topics are mentioned below.

5.1 General multiswarm P2P models

Theorem 3.1.1 based on the case in which the seed applies random novel piece selection. Intuition on the missing piece syndrome and simulations both suggest that the stability region may be the same if work-conserving piece selections are applied by the seed. However, the proof of positive recurrence provided in Section 3.7 does not work if seeds apply any work-conserving piece selection policy. A careful study and comparison of piece selection policies on seeds may lead to counter-intuitive results on scalability.

Methods like peers arriving with pieces, peer seeds dwelling, and network coding can be merged into the multiswarm models, like that in Chapter 2. More general stability regions can be generated like those in Theorems 2.1.1

and 2.7.1, which can reveal more intrinsic properties of multiswarm networks.

5.2 Peer selections and non-work-conserving piece selections

In this thesis uniform sampling is assumed on a complete underlying graph. The tit-for-tat algorithm applied in BitTorrent is not covered in the random sampling model. Analyzing the effects of diverse peer selection policies remains to be done. Intuitively, it may enlarge the stability region if peers with rare pieces can be located.

Studying peer selection policies on multiswarm networks is also an interesting topic. It remains unknown to determine how the way that a peer's allocation of its capacity, in its own swarm or in other swarms, affects the performance. One conjecture here is that if capacities of peers are allocated in a static way across all swarms, the stability region in Theorem 3.1.1 still holds.

Another interesting topic is to study the effects of non-work-conserving piece selection. In [16] the P2P network is shown to be always stable under a non-work-conserving piece selection policy. Intuitively, peers stay longer and contribute more if piece selections are non-work-conserving, which can enlarge the stability region. Increase in sojourn time caused by non-work-conserving policies may be negligible in systems with smarter designs.

5.3 Improving the metric of stability

Metastability of the rarest first piece selection shown in Section 3.3 suggests that positive recurrence is not the best metric for stability of P2P networks. The reason is that with rarest first piece selection, transient systems can work well under normal states for tens of years or longer before getting into states of the missing piece syndrome. That is, the probability for the missing piece syndrome to manifest is so low that it can be neglected, for systems under rarest first piece selection policy. Though systems like that may be transient, they can actually be considered as stable systems. However, metastability is not a well defined status. It remains as an open problem to find a better

metric for the stability of P2P networks, which can distinguish metastability from instability.

5.4 Tree management algorithms on fixed topologies

In Chapter 4 the P2P streaming model is built on the assumption of random peer sampling on a complete underlying topology, which is true of the models in Chapters 2 and 3 as well. One drawback of the assumption is that it cannot capture heterogeneous link bandwidth between peers. It remains open whether the algorithm in Chapter 4 can be extended to a fixed underlying topology with given link capacities, like that in [36]. The difficulty in extending the algorithm is that the problem usually is NP hard to identify the best multicast trees on a given graph. But good approximations may be generated on special graphs.

Appendix A

MISCELLANEOUS RESULTS ON STOCHASTIC PROCESS

A.1 Foster-Lyapunov stability criterion

Proposition A.1.1 *(See [54, 57, 59].) Suppose X is a continuous-time, irreducible Markov process on a countable state space $\mathcal{S}$ with generator matrix Q. Suppose V, f, and g are nonnegative functions on $\mathcal{S}$ such that $QV(\mathbf{x}) \leq -f(\mathbf{x}) + g(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{S}$, and, for some $\delta > 0$, the set C defined by $C = \{\mathbf{x} : f(\mathbf{x}) < g(\mathbf{x}) + \delta\}$ is finite. Suppose also that $\{\mathbf{x} : V(\mathbf{x}) \leq K\}$ is finite for all K. Then X is positive recurrent and, if π denotes the equilibrium distribution, $\sum_{\mathbf{x}} f(\mathbf{x})\pi(\mathbf{x}) \leq \sum_{\mathbf{x}} g(\mathbf{x})\pi(\mathbf{x})$.*

A.2 Bounding the drift of functions

Lemma A.2.1 *Suppose X is a continuous-time, irreducible Markov process with countable state space $\mathcal{S}$ and with generator matrix $Q = (q(x, x'), x, x' \in \mathcal{S})$. Suppose $f : \mathcal{S} \to [0, \infty)$ and $V : \mathbb{R} \to [0, \infty)$ are two nonnegative functions; and suppose V is continuously differentiable and has second derivative existing almost everywhere and bounded above by M almost everywhere. Then $QV(f)$, the drift of $V(f)$, satisfies*

$$\begin{aligned} QV(f)(z) &:= \sum_{z' : z' \in \mathcal{S}, z' \neq z} q(z, z') \left[V(f(z')) - V(f(z)) \right] \\ &\leq V'(f(z)) Qf(z) + \frac{M}{2} \sum_{z' : z' \in \mathcal{S}, z' \neq z} q(z, z') [f(z') - f(z)]^2 \end{aligned}$$

where $z \in \mathcal{S}$ and Qf is the drift of f.

Proof. The lemma follows from the following:

$$\begin{aligned} V(f(z')) - V(f(z)) &= \int_{f(z)}^{f(z')} V'(x)dx = \int_{f(z)}^{f(z')} \left[V'(z) + \int_{f(z)}^{x} V''(y)dy \right] dx \\ &\leq \int_{f(z)}^{f(z')} \left[V'(z) + M(x - f(z)) \right] dx \\ &= V'(z)[f(z') - f(z)] + \frac{M}{2}[f(z') - f(z)]^2. \end{aligned}$$

□

A.3 Stochastic domination or coupling

Definition A.3.1 *Suppose $A = (A_t : t \geq 0)$ and $B = (B_t : t \geq 0)$ are two random processes, either both discrete-time random processes, or both continuous time random processes having right-continuous with left limits sample paths. Then A is* stochastically dominated *by B if there is a single probability space $(\Omega, \mathcal{F}, P)$, and two random processes $\tilde{A}$ and $\tilde{B}$ on $(\Omega, \mathcal{F}, P)$, such that*

(a) A and $\tilde{A}$ have the same finite dimensional distributions,

(b) B and $\tilde{B}$ have the same finite dimensional distributions, and

(c) $P\{\tilde{A}_t \leq \tilde{B}_t \text{ for all } t\} = 1$.

Clearly if A is stochastically dominated by B, then for any a and t, $P\{A_t \geq a\} \leq P\{B_t \geq a\}$.

A.4 Kingman's moment bound

Proposition A.4.1 *([60], see [12] for proof) Let C be a compound Poisson process with $C_0 = 0$, with jumps times given by a Poisson process of rate α, and jump sizes having mean m_1 and mean square value m_2. Then for all $B > 0$ and $\epsilon > \alpha m_1$*

$$P\{C_t < B + \epsilon t \text{ for all } t\} \geq 1 - \frac{\alpha m_2}{2B(\epsilon - \alpha m_1)} \tag{A.1}$$

A.5 Busy periods for $\mathbf{M}/GI/1$ queues

Consider an $M/GI/1$ queue with arrival rate λ. Let N denote the number of customers served in a busy period, let L denote the length of a busy period, and let X denote the service time of a typical customer.

Lemma A.5.1 *(See [12] for proof) Let $\rho = E[X]$. It $\rho < 1$ then*

$$E[N] = \frac{1}{1-\rho}, \quad E[N^2] = \frac{1+\lambda^2 Var(X)}{(1-\rho)^3}$$
$$E[L] = \frac{E[X]}{1-\rho}, \quad E[L^2] = \frac{E[X^2]}{(1-\rho)^3}$$
$$Cov(N, L) = \frac{\lambda E[X^2]}{(1-\rho)^3}.$$

A.6 A bound for $\mathbf{M}/GI/\infty$ queues

Lemma A.6.1 *(See [12] for proof) Let M denote the number of customers in an $M/GI/\infty$ queueing system, with arrival rate λ and mean service time m. Suppose that $M_0 = 0$. Then for $B, \epsilon > 0$,*

$$P\{M_t \geq B + \epsilon t \ \text{ for some } t \geq 0\} \leq \frac{e^{\lambda(m+1)} 2^{-B}}{1 - 2^{-\epsilon}} \tag{A.2}$$

Appendix B

PROOFS IN CHAPTER 2

B.1 Proof of Lemma 2.4.1

We describe a particular method of coupling the ABS and the original system. By this, we mean a way to construct both processes on a single probability space. To do this, we start with the random variables governing the ABS, and describe how the original system (i.e. a system with the statistical description of the original system) can be overlaid on the same probability space, in such a way that $D_t \leq \widehat{D}_t$ for all t with probability one.

The first step is to adopt a new way of thinking about the ABS. In the ABS, the sets of descendants of the root peers form a partition of all group (b) and group (f) peers in the ABS (for this purpose, the descendants of a root peer include the root peer itself if the root peer is an offspring of the fixed seed, but not if the root peer is a gifted peer). Imagine that each root peer arrives with a randomly generated *script* for itself and its descendants. The script includes the sample paths of the Poisson processes that determine: when pieces are to be downloaded, when group (b) peers are spawned, and when group (f) peers are spawned, as well as the seed dwell durations sampled from the exponential distribution with parameter γ. Whenever some peer in the ABS system spawns another, the portion of the script held by the parent associated with that offspring and its descendants becomes a script for that offspring.

The next step is to build the original system using the same random variables, using the following assumptions. When thinning of Poisson processes is mentioned, it refers to randomly rejecting some points of a Poisson process to produce another point process with a specified intensity that is smaller than the rate of the Poisson process.

1. The original system has independent Poisson arrivals of peers of type

C at rate λ_C, for all C with $1 \notin C$. These arrivals are not modeled in the ABS, and are to be generated for the original system independently of the ABS.

2. The arrival processes of gifted peers of type C in the original system for all C with $1 \in C$ are identical to those in the ABS.

3. The point process of times that the seed uploads piece one to one-club peers is a thinning of the rate U Poisson process governing creation of group (f) peers in the ABS system.

4. The point process of times that the seed uploads piece one to normal young peers is a thinning of the rate ξU Poisson process governing creation of group (b) peers in the ABS system.

5. The point process of times that a peer in group (b), (g), or (f) uploads piece one to one-club peers is a thinning of the rate μ Poisson process in the script of the peer for spawning group (f) peers.

6. The point process of times that a peer in group (b), (g), or (f) uploads piece one to normal young peers is a thinning of the rate $\xi\mu$ Poisson process in the script of the peer for spawning group (b) peers.

7. A peer in the original system in group (b) or (g) that does not have a complete collection downloads useful pieces from a peer in group (e) or (f) at the jump times of the rate $\mu(1-\xi)$ Poisson process for downloads in its script. The peer can also make downloads at other times, to bring the total intensity of downloads from groups (e) and (f) up to at least $\frac{\mu(Y^e+Y^f)}{N}$.

8. The peer seed dwell time for any peer is specified in its script.

A remark is in order about why the construction is possible. When one peer transfers a piece to another peer in the original system, it is considered an upload for the first peer and a download for the second. Thus, the timing of such transfers cannot be simultaneously governed by internal scripts of the two peers. In the construction noted here, such conflict does not occur, because the scripts are used to determine times that piece one can be uploaded, and the peers that are downloading piece one are in group (a) or (e), and

are thus not yet following a script. And the scripts are used for downloading of pieces other than piece one, but do not constrain times that pieces other than piece one are uploaded.

The resulting coupling satisfies the following properties.

- Any peer in group (b), (f), or (g) in the original system is also in the ABS, in the same group and with the same time of arrival to that group. (Such peers can remain in the ABS longer than they stay in the original system.)
- Any peer in group (f) in the original system (and thus also in the ABS) departs from both systems at the same time. (Peers in groups (b) or (g) in the original system can stay longer in the ABS than in the original system.)
- Whenever some peer p_1 in the original system uploads piece one to some other peer p_2, peer p_1 simultaneously spawns peer p_2 in the ABS. Afterwards, peer p_2 is either in group (b) or in group (f) in both systems.
- There can be more group (b) and more group (f) peers in the ABS than in the original system because the spawning rates in the ABS system are greater than in the original system, and group (b) and group (f) peers in the ABS can have fewer pieces in the ABS system than they have in the original system, and thus they can stay longer in the ABS system than in the original system.

In particular, by the third point above, whenever piece one is uploaded in the original system a peer of group (b) or (f) is created in the ABS system. Therefore, $D_t \leq \widehat{D}_t$ for all $t \geq 0$ with probability one, which by the definition of stochastic domination, proves the lemma.

B.2 Proof of Corollary 2.4.1

By Lemma 3.6.4, it suffices to prove Corollary 2.4.1 with D replaced by $\widehat{D}$. Let $\widetilde{D}$ be a random process associated with the ABS, denoting the cumulative counting process that results if all the descendants of a root peer are counted

at the time the root peer arrives. The processes $\widehat{D}$ and $\widetilde{D}$ count the same downloads of piece one, but $\widetilde{D}$ does so sooner, so $\widehat{D}_t \leq \widetilde{D}_t$ for all t. Thus, it suffices to prove Corollary 2.4.1 with D replaced by $\widetilde{D}$. The process $\widetilde{D}$ is a compound Poisson process, which can be decomposed into the sum of several independent compound Poisson processes: one for each type C with $1 \in C$, and one for peer seeds generated directly by the fixed seed. The mean arrival rate for $\widetilde{D}$ satisfies:

$$\begin{aligned} \frac{E\left[\widetilde{D}_t\right]}{t} &= U(\xi m_b + m_f) + \sum_{C:1\in C} \lambda_C m_g(C) \\ &\stackrel{\xi\to 0}{\to} \left[U + \sum_{C:k\in C} \lambda_C \left(K - |C| + \frac{\mu}{\gamma}\right)\right] \left(\frac{1}{1-\frac{\mu}{\gamma}}\right) \end{aligned}$$

and the batch sizes have finite second moments for ξ sufficiently small, and the second moments are increasing in ξ. Therefore, Corollary 2.4.1 follows from Kingman's moment bound (see Proposition A.4.1 in the appendix).

B.3 Proof of Lemma 2.4.3

The idea of the proof is to show how, with a possible enlargement of the underlying probability space, an $M/GI/\infty$ system can be constructed on the same probability space as the original system, so that for any time t, $Y_t^a + Y_t^b + Y_t^g$ is less than or equal to the number of peers in the $M/GI/\infty$ system. Let the $M/GI/\infty$ system have the same arrival process as the original system–it is a Poisson process of rate λ_{total}.

An important point is that any peer in group (a), (b), or (g) is either receiving useful download opportunities at rate at least $(1-\xi)\mu$, or is a peer seed (possible if it is in group (b) or (g)) and is thus waiting for a departure time that is exponentially distributed with parameter γ. We can thus imagine that any arriving peer has an internal Poisson clock that ticks at rate $\mu(1-\xi)$ and an internal, exponentially distributed random variable with parameter γ. Whenever its internal clock ticks, it can download a useful piece, until it either joins the one-club (in which case it leaves group (a) and joins group (e)) or it becomes a peer seed, in which case it remains in the system as a peer seed for an amount of time equal to its internal exponential random

variable of parameter γ.

An arriving peer in the original system may already have some pieces at the time of arrival, or its intensity of downloading pieces could be greater than $(1-\xi)\mu$, or it might leave the union of groups (a), (b), and (g) by becoming a one-club peer. These factors reduce the time that a peer remains in the union of groups (a), (b) and (g). The $M/GI/\infty$ system is constructed by ignoring those speedup factors. Specifically, in the $M/GI/\infty$ system, each arriving peer has to download K pieces at times governed by its internal Poisson clock, and then remain as a peer seed for a time duration given by its internal exponentially distributed random variable for seed time. The service time distribution for the $M/GI/\infty$ system is thus the sum of K independent exponential random variables with parameter $\mu(1-\xi)$ plus a single exponential random variable with parameter γ. Any peer that is in groups (a), (b), or (g) in the original system will be in the $M/GI/\infty$ system, and the mean service time for the $M/GI/\infty$ system is $\frac{K}{\mu(1-\xi)} + \frac{1}{\gamma}$.

B.4 Proof of Lemma 2.5.2

Compare $Q(W)$ and $ŁW$ term by term. Consider terms of the form $Q(T_C)$ and $ŁT_C$. First assume $C \neq \mathcal{F}$. Because $\alpha < 1$, we can write

$$
\begin{aligned}
&|Q(T_C) - ŁT_C| \leq a_1 + a_2 + a_3 \text{ where} \\
&a_1 = \left|\frac{1}{2}Q(E_C^2) - E_C Q(E_C)\right| = \lambda_{\mathcal{E}_C} + \Gamma_{\mathcal{E}_C,\mathcal{H}_C} \leq \lambda_{total} + D_{total} \\
&a_2 = \left|Q(E_C\phi(H_C)) - \left[Q(E_C)\phi(H_C) + E_C Q(\phi(H_C))\right]\right| \\
&a_3 = |Q(E_C)\phi(H_C)| \leq M_\phi(\lambda_{\mathcal{E}_C} + \Gamma_{\mathcal{E}_C,\mathcal{H}_C})
\end{aligned}
$$

The only way E_C and $\phi(H_C)$ can simultaneously change is that some peer with type in $\mathcal{E}_C$ becomes a peer with type in $\mathcal{H}_C$, causing E_C to decrease by 1, and $\phi(H_C)$ to decrease by at most $\frac{K+\mu/\gamma}{1-\mu/\gamma}$, so

$$
a_2 \leq \frac{K+\mu/\gamma}{1-\mu/\gamma}\Gamma_{\mathcal{E}_C,\mathcal{H}_C}
$$

From the discussion above and the fact that $\Gamma_{\mathcal{E}_C,\mathcal{H}_C} \leq D_{total}$, we have

$$|Q(T_C) - \text{Ł}T_C| \leq M_\phi\Theta(1) + M_\phi D_{total}\Theta(1) \in M_\phi(D_{total}+1)\Theta(1) \quad \text{(B.1)}$$

for every $C \in \mathcal{C} \setminus \{\mathcal{F}\}$.

Second, assume $C = \mathcal{F}$ and $\gamma < \infty$. Then,

$$|Q(T_C) - \text{Ł}T_C| = \left|\frac{1}{2}Q(n^2) - nQ(n)\right| = \lambda_{total} + D_{\mathcal{F}} \leq \lambda_{total} + D_{total},$$

which implies (B.1) for $C = \mathcal{F}$. There are only finitely many terms of T_C in W (2^K in total), and $r < 1$: Lemma 2.5.2 follows.

B.5 Proof of Lemma 2.5.3

The upper bound for the drift of E_C is obvious: $Q(E_C) \leq \lambda_{\mathcal{E}_C} \leq \lambda_{total}$. Next consider $Q(\phi(H_C))$. Since $H_{\mathcal{F}} \equiv 0$, we restrict our attention to the case $C \neq \mathcal{F}$. Because ϕ is a decreasing function, only the rate for H_C to decrease contributes to the positive part in the drift of $\phi(H_C)$, so to consider an upper bound of $Q(\phi(H_C))$ it suffices to consider the rates of transitions that decrease H_C. There are two ways H_C can decrease: peers with one type in $\mathcal{H}_C$ becoming another type in $\mathcal{H}_C$ – with aggregate rate $\Gamma_{\mathcal{H}_C,\mathcal{H}_C}$, and peer seeds departing – with rate $D_{\mathcal{F}}$. Because the maximum that $\phi(H_C)$ can jump up is less than or equal to $\frac{1+\mu/\gamma}{1-\mu/\gamma}$, an upper bound for the drift of $\phi(H_C)$ is

$$\begin{aligned} Q(\phi(H_C)) &\leq \frac{1+\mu/\gamma}{1-\mu/\gamma}(\Gamma_{\mathcal{H}_C,\mathcal{H}_C} + D_{\mathcal{F}}) \leq \frac{1+\mu/\gamma}{1-\mu/\gamma}\left[U + H_C\left(\mu + \gamma\mathbf{1}_{\{\gamma<\infty\}}\right)\right] \\ &\in \Theta(1) + H_C\Theta(1) \end{aligned}$$

We can choose d large enough, i.e. $d > \frac{1+\mu/\gamma}{1-\mu/\gamma}$, so $M_\phi > 2d + 1/\beta + \frac{1+\mu/\gamma}{1-\mu/\gamma}$. Thus $Q(\phi(H_C))$ vanishes when $H_C > M_\phi$, because $\phi(H_C)$ vanishes when $H_C > 2d + 1/\beta$ and the jump size of H_C is bounded below by $-\frac{1+\mu/\gamma}{1-\mu/\gamma} \geq -d$. Hence

$$Q(\phi(H_C)) \leq \Theta(1) + M_\phi\Theta(1) \in M_\phi\Theta(1),$$

since $M_\phi > 1$.

Finally, the bound on $\text{Ł}T_C$ follows from the other two bounds already

proved. Hence, Lemma 2.5.3 is proved.

B.6 Proof of Lemma 2.5.4

Now the detailed proof of Lemma 2.5.4 is given. Consider a nonzero state $\mathbf{n}$ of type I, with $S \in \mathcal{C} \setminus \{\mathcal{F}\}$, $n_S/n > 1 - \epsilon$. Recall that (2.4) is assumed to hold; $\triangle_S < 0$. We begin with three observations.

First, consider a lower bound for $Q(H_S)$:

$$\begin{aligned} Q(H_S) &= \frac{1}{1-\mu/\gamma} \sum_{C':C' \in \mathcal{H}_S} (K - |C'| + \mu/\gamma) Q(n_{C'}) \\ &\geq \frac{1}{1-\mu/\gamma} \left[\lambda^*_{\mathcal{H}_S} + D_S \frac{\mu}{\gamma} - \Gamma_{\mathcal{H}_S,\mathcal{H}_S} - D_{\mathcal{F}} \frac{\mu}{\gamma} \right] \\ &= \frac{1}{1-\mu/\gamma} \left(\lambda^*_{\mathcal{H}_S} + b_1 \right) - D_S, \end{aligned} \tag{B.2}$$

where $b_1 := D_S - \Gamma_{\mathcal{H}_S,\mathcal{H}_S} - n_{\mathcal{F}}\mu$. Since

$$D_S \geq (1-\epsilon)(U + n_{\mathcal{H}_S}\mu) \geq U + n_{\mathcal{H}_S}\mu - \epsilon[\Theta(1) + n_{\mathcal{H}_S}\Theta(1)] \tag{B.3}$$

$$\Gamma_{\mathcal{H}_S,\mathcal{H}_S} + n_{\mathcal{F}}\mu \leq \epsilon U + n_{\mathcal{H}_S}\mu \tag{B.4}$$

it follows that

$$b_1 \geq U - \epsilon[\Theta(1) + n_{\mathcal{H}_S}\Theta(1)]. \tag{B.5}$$

Combining (B.2) with (B.5), yields:

$$Q(H_S) \geq -h_1 - \epsilon[\Theta(1) + n_{\mathcal{H}_S}\Theta(1)], \tag{B.6}$$

$$h_1 := D_S - \frac{1}{1-\mu/\gamma} \left(\lambda^*_{\mathcal{H}_S} + U \right) \tag{B.7}$$

Second,

$$D_S \geq n_{\mathcal{H}_S}\mu(1-\epsilon) = n_{\mathcal{H}_S}\Theta(1) = H_S\Theta(1), \tag{B.8}$$

because $n_{\mathcal{H}_S} \leq H_S \leq \frac{K+\mu/\gamma}{1-\mu/\gamma} n_{\mathcal{H}_S}$ and $\epsilon < \frac{1}{2}$.

Third, substituting (B.8) into (B.7) yields that if d is sufficiently large, then $h_1 \geq d\Theta(1) - \Theta(1)$ whenever $H_S > d$. Therefore, if d is sufficiently

large,

$$h_1 > 0 \quad \text{whenever} \quad H_S > d. \tag{B.9}$$

The remainder of the proof is divided into two, according to the value of H_S.

- $\mathbf{H_S \leq M_\phi}$: Under this condition, $n_{\mathcal{H}_S} \leq M_\phi$ and $M_\phi > 1$, so (B.6) implies:

$$Q(H_S) \geq -h_1 - \epsilon M_\phi \Theta(1). \tag{B.10}$$

Because $\phi'' \leq \beta$ at points where the second derivative ϕ'' exists, and because the magnitudes of the jumps of H_S are bounded by $\frac{K+\mu/\gamma}{1-\mu/\gamma}$, Lemma A.2.1 yields

$$Q(\phi(H_S)) \leq \phi'(H_S)Q(H_S) + b_2 \tag{B.11}$$

$$b_2 := \frac{\beta}{2}\left(\frac{K+\mu/\gamma}{1-\mu/\gamma}\right)^2 (\lambda_{\mathcal{H}_S} + \Gamma_{\mathcal{E}_S,\mathcal{H}_S} + \Gamma_{\mathcal{H}_S,\mathcal{H}_S} + n_{\mathcal{F}}\mu) \tag{B.12}$$

Upper bounds for the terms in the right-hand side of (B.11) are found next. First, a bound for b_2 is found. By (B.3) and (B.4),

$$\Gamma_{\mathcal{E}_S,\mathcal{H}_S} \leq D_S + \epsilon(U + n_{\mathcal{H}_S}\mu) \leq D_S + \epsilon M_\phi \Theta(1) \tag{B.13}$$

$$\Gamma_{\mathcal{H}_S,\mathcal{H}_S} + n_{\mathcal{F}}\mu \leq U + n_{\mathcal{H}_S}\mu \leq D_S + \epsilon M_\phi \Theta(1) \tag{B.14}$$

Substituting (B.13) and (B.14) into the right side of (B.12) yields

$$b_2 \leq \beta\Theta(1) + \beta\epsilon M_\phi \Theta(1) + \beta\left(\frac{K+\mu/\gamma}{1-\mu/\gamma}\right)^2 D_S \tag{B.15}$$

$$\leq \left(\frac{1}{\alpha} - 1\right) D_S + \beta\Theta(1), \tag{B.16}$$

where to obtain (B.16) from (B.15), we assume $\beta\left(\frac{K+\mu/\gamma}{1-\mu/\gamma}\right)^2 \leq \frac{1}{\alpha} - 1$ and $\epsilon M_\phi < 1$.

Second, a bound for $\phi'(H_S)Q(H_S)$ is found. Taking into account that $-1 \leq \phi' \leq 0$, multiply both sides of (B.10) by $\phi'(H_S)$ and use the fact

$\phi'(H_s) = -1$ for $H_s \leq d$, and (B.9), to obtain:

$$\phi'(H_S)Q(H_S) \leq -\phi'(H_S)h_1 + \epsilon M_\phi \Theta(1) \leq h_1 + \epsilon M_\phi \Theta(1) \quad \text{(B.17)}$$

Substituting (B.7), (B.16) and (B.17) into (B.11) yields

$$Q(\phi(H_S)) \leq \frac{1}{\alpha} D_s - \frac{1}{1-\mu/\gamma} \left(\lambda^*_{\mathcal{H}_S} + U\right) + \epsilon M_\phi \Theta(1) + \beta\Theta(1). \quad \text{(B.18)}$$

We obtain a bound on $Q(E_S) + \alpha Q(\phi(H_S))$, the coefficient of E_S in $\text{Ł}T_S$, using (B.18) and the facts $Q(E_S) \leq \lambda_{\mathcal{E}_S} - D_S$ and $\alpha < 1$, as follows:

$$\begin{aligned} Q(E_S) + \alpha Q(\phi(H_S)) &\leq \lambda_{\mathcal{E}_S} - \frac{\alpha}{1-\mu/\gamma} \left(\lambda^*_{\mathcal{H}_S} + U\right) + \epsilon M_\phi \Theta(1) + \beta\Theta(1) \\ &\leq \triangle_S + (1-\alpha)\Theta(1) + \epsilon M_\phi \Theta(1) + \beta\Theta(1) \quad \text{(B.19)} \end{aligned}$$

Because $\triangle_S < 0$, if $1-\alpha, \epsilon M_\phi, \beta$ are close to 0, the last three terms in (B.19) can be made small compared to $|\triangle_S|$, so $Q(E_S) + \alpha Q(\phi(H_S)) \leq \frac{1}{2}\triangle_S$, which implies (2.16).

- $\mathbf{H_S > M_\phi}$: To take care of this case, assume $d > \frac{K+\mu/\gamma}{1-\mu/\gamma}$, so $M_\phi > 2d + 1/\beta + \frac{K+\mu/\gamma}{1-\mu/\gamma}$. Hence $Q(\phi(H_S))$ vanishes for H_S in this range. By (B.8),

$$Q(E_S) + \alpha Q(\phi(H_S)) \leq \lambda_{\mathcal{E}_S} - D_S \leq \Theta(1) - M_\phi\Theta(1) < \frac{1}{2}\triangle_S, \quad \text{(B.20)}$$

if d is large enough so that M_ϕ is large enough. Therefore (2.16) holds.

The proof of Lemma 2.5.4 is now complete.

B.7 Proof of Lemma 2.5.5

First consider Lemma 2.5.5(a). Since there are only finitely many types, we can fix a set $S \in \mathcal{C} \setminus \{\mathcal{F}\}$ and consider the set of class I states $\mathbf{n}$ for which $n_S/n > 1 - \epsilon$. Since $\epsilon \in (0, \frac{1}{2})$, $E_S > \frac{1}{2}n$. By assumption in this section,

$\triangle_S < 0$. By Lemma 2.5.4,

$$Ł T_S \leq \frac{1}{4}\triangle_S n \in -\Theta(n). \tag{B.21}$$

For type C with $|C| > |S|$, Lemma 2.5.3 and (C.4) imply

$$r^{|C|}Ł T_C \leq r M_\phi r^{|S|}\Theta(n) < 2^{-K-1} r^{|S|}\big|Ł T_S\big| \tag{B.22}$$

if rM_ϕ is chosen to be small enough.

For type C with $|C| \leq |S|$ but $C \neq S$, $E_C \leq \epsilon n$; Lemma 2.5.3 and (C.4) imply

$$r^{|C|}Ł T_C \leq r^{|C|} M_\phi \Theta(E_C) \leq \epsilon M_\phi r^{-K} r^{|S|}\Theta(n) < 2^{-K-1} r^{|S|}\big|Ł T_S\big| \tag{B.23}$$

if $\epsilon M_\phi r^{-K}$ is chosen to be small enough.

Equations (C.6) and (C.6) imply that

$$\begin{aligned} Ł W &= r^{|S|}Ł T_S + \sum_{C:|C|>|S|} r^{|C|}Ł T_C + \sum_{C:|C|\leq|S|, C\neq S} r^{|C|}Ł T_C \\ &\leq r^{|S|}Ł T_S + \frac{1}{2} r^{|S|}\big|Ł T_S\big| \leq \frac{1}{8} r^{|S|}\triangle_S n \leq -r^K \Theta(n), \end{aligned}$$

which proves Lemma 2.5.5(a).

Next consider Lemma 2.5.5(b). First, suppose $C_1 \not\subseteq C_2$ and consider the set of class II states $\mathbf{n}$ such that $n_{C_1}/n > \eta, n_{C_2}/n > \eta$, where $\eta = \epsilon/2^K$. For such states:

$$\Gamma_{\mathcal{E}_{C_2},\mathcal{H}_{C_2}} \geq D_{C_2} \geq \frac{n_{C_2}}{n} n_{C_1}\mu \geq \mu\eta^2 n \in \epsilon^2\Theta(n). \tag{B.24}$$

Since $E_{C_2} \geq n_{C_2} \geq \eta n$, (C.6) implies

$$E_{C_2} Q(E_{C_2}) = E_{C_2}(\lambda_{\mathcal{E}_{C_2}} - \Gamma_{\mathcal{E}_{C_2},\mathcal{H}_{C_2}}) \leq -\epsilon^3\Theta(n^2) + \Theta(n). \tag{B.25}$$

Lemma 2.5.3 indicates that $E_{C_2} Q(\phi(H_{C_2})) \leq M_\phi \Theta(n)$, so (C.6) implies

$$Ł T_{C_2} = E_{C_2} Q(E_{C_2}) + \alpha E_{C_2} Q(\phi(H_{C_2})) \leq -\epsilon^3\Theta(n^2) + M_\phi\Theta(n). \tag{B.26}$$

Second, consider the set of class II states $\mathbf{n}$ such that $n_{\mathcal{F}}/n > \eta$, where

$\eta = \epsilon/2^K$. If $\gamma = \infty$, this set is empty, so suppose $\gamma < \infty$. For such states,

$$\begin{aligned} \text{Ł}T_{\mathcal{F}} &= nQ(n) = n(\lambda_{total} - \gamma n_{\mathcal{F}}) \\ &\leq n(\lambda_{total} - \eta\gamma n) \in -\epsilon\Theta(n^2) + \Theta(n). \end{aligned} \tag{B.27}$$

Recall that Lemma 2.5.3 implies for any C, $\text{Ł}T_C \leq M_\phi\Theta(n)$. Therefore, for either condition $C_1 \not\subseteq C_2$ or $C_1 = C_2 = \mathcal{F}$, (C.6) and (B.27) imply that, over the set of all class II states,

$$\text{Ł}W \leq r^{|C_2|}\text{Ł}T_{C_2} + \sum_{C:C\neq C_2} \text{Ł}T_C \leq -r^K\epsilon^3\Theta(n^2) + M_\phi\Theta(n),$$

which proves Lemma 2.5.5(b).

B.8 Proof of Lemma 2.5.6

Suppose $S \in \mathcal{C} \setminus \{\mathcal{F}\}$ and $n_S/n > 1 - \epsilon$, $\epsilon \in (0, \frac{1}{2})$, one lower bound for $Q(H'_S)$ is:

$$Q(H'_S) \geq \lambda^*_{\mathcal{H}_S} + D_S - \Gamma_{\mathcal{H}_S,\mathcal{H}_S} - n_{\mathcal{F}}\gamma \geq \lambda^*_{\mathcal{H}_S} + b_1 \tag{B.28}$$

since $\lambda \leq \mu$. Substituting (B.5) into (B.28),

$$Q(H'_S) \geq \lambda^*_{\mathcal{H}_S} + U - \epsilon[\Theta(1) + n_{\mathcal{H}_S}\Theta(1)] \tag{B.29}$$

Consider two conditions of H'_S:

- $\mathbf{H'_S \leq M_\phi}$: Under this condition, $n_{\mathcal{H}_S} \leq M_\phi$ and $M_\phi > 1$, so (B.29) becomes:

$$Q(H'_S) \geq U + \lambda^*_{\mathcal{H}_S} - \epsilon M_\phi\Theta(1). \tag{B.30}$$

  Because $\phi'' \leq \beta$ at points where the second derivative ϕ'' exists, and because the magnitude of the jump of H'_S is bounded by $K + 1$, by

Lemma A.2.1,

$$Q(\phi(H'_S)) \leq \phi'(H'_S)Q(H'_S) + b'_2 \tag{B.31}$$

$$b'_2 := \frac{\beta}{2}(K+1)^2(\lambda_{\mathcal{H}_S} + \Gamma_{\mathcal{E}_S,\mathcal{H}_S} + \Gamma_{\mathcal{H}_S,\mathcal{H}_S} + n_{\mathcal{F}}\mu) \tag{B.32}$$

Consider the term b'_2. By (B.13) and (B.14), and assume $\epsilon M_\phi < 1$,

$$\begin{aligned} b'_2 &\leq \beta\Theta(1) + \beta\epsilon M_\phi\Theta(1) + \beta D_S\Theta(1) \\ &\leq \beta D_S\Theta(1) + \beta\Theta(1). \end{aligned} \tag{B.33}$$

If β is small enough, $\beta D_S\Theta(1) < \frac{1}{2p}D_S$, so (B.33) becomes:

$$b'_2 \leq \frac{1}{2p}D_S + \beta\Theta(1). \tag{B.34}$$

Substituting (B.30) and (B.34) into (B.31), and applying $Q(E_S) \leq \lambda_{\mathcal{E}_S} - D_S$, we can bound $Q(E_S) + pQ(\phi(H'_S))$, the coefficient of E_S in $\text{Ł}T'_S$, as follows:

$$\begin{aligned} & Q(E_S) + pQ(\phi(H'_S)) \\ \leq\ & \lambda_{\mathcal{E}_S} - \frac{1}{2}D_S + p\phi'(H'_S)(U + \lambda^*_{\mathcal{H}_S}) + \beta\Theta(1) + \epsilon M_\phi\Theta(1) \\ =\ & \lambda_{\mathcal{E}_S} - p(U + \lambda^*_{\mathcal{H}_S}) + b'_3 + \beta\Theta(1) + \epsilon M_\phi\Theta(1), \end{aligned} \tag{B.35}$$

where

$$b'_3 := p(1 + \phi'(H'_S))(U + \lambda^*_{\mathcal{H}_S}) - \frac{1}{2}D_S \tag{B.36}$$

$$\leq \begin{cases} -\frac{1}{2}D_S & \text{if } H'_S < d \\ \Theta(1) - d\Theta(1) & \text{if } H'_S \geq d \end{cases} \tag{B.37}$$

because $D_S \geq (1-\epsilon)n_{\mathcal{H}_S}\mu \geq \frac{1}{2(K+1)}H'_S\mu \in \Theta(H'_S)$. Hence if d is large enough, $b'_3 \leq 0$. If $\beta, \epsilon M_\phi$ are close to 0, the last two terms in (B.35) can be neglected. Thus, $Q(E_S) + pQ(\phi(H'_S)) \leq \frac{1}{2}[\lambda_{\mathcal{E}_S} - p(U + \lambda^*_{\mathcal{H}_S})]$, which implies (2.22).

- $\mathbf{H'_S > M_\phi}$: Under this condition, choose d such that $d > K+1$, so $M_\phi > 2d+1/\beta+K+1$. Hence $Q(\phi(H'_S))$ vanishes for H'_S in this range.

The fact that $D_S \in \Theta(H'_S)$ yields that

$$\begin{aligned} Q(E_S) + pQ(\phi(H'_S)) &\leq \lambda_{\mathcal{E}_S} - D_S \\ &\leq \Theta(1) - M_\phi \Theta(1) < \frac{1}{2}[\lambda_{\mathcal{E}_S} - p(U + \lambda^*_{\mathcal{H}_S})], \end{aligned}$$

if d is large enough, and hence also M_ϕ. Therefore (2.22) holds.

So far, Lemma 2.5.6 is proved.

Appendix C

PROOFS IN CHAPTER 3

C.1 Proof of Lemma 3.6.3

We construct one $M/GI/\infty$ queue on the same probability space as the alternative system, so that $Y_t \leq$ the number of peers in the $M/GI/\infty$ queue. Let the $M/GI/\infty$ queue have the same arrival process as the alternative system–*i.e.*, a Poisson process of rate λ_{total}. As $\xi < 0.5$, for any young peer, the intensity of downloads from the one-club is always greater than or equal to $\mu/2$ for the alternative system. Suppose thus that each young peer has an internal Poisson clock, which ticks at rate $\mu/2$, and is such that whenever the internal clock of a young peer ticks, that young peer downloads a piece from the one-club. We declare that a peer remains in the $M/GI/\infty$ system until its internal clock ticks $K-1$ times. This gives the desired service time distribution, and the service times of different peers in the $M/GI/\infty$ are independent. A young peer may leave the group of young peers (depart or join the one-club) sooner than it leaves the $M/GI/\infty$ system, because a young peer in the alternative system can possibly download pieces at times when its internal clock does not tick. But if a peer is still a young peer in the alternative system, it is in the $M/GI/\infty$ system.

C.2 Proof of Lemma 3.6.5

Identify two kinds of infected peers in the comparison system–the *root peers*, which are those created by the seed, and the infected peers created by other infected peers. Assume each root peer signs uniquely on its piece one received from the seed, and the signature is inherited by all copies of piece one replicated from the root peer. Partition the uploads of piece one according

to their signatures. Let $(\widetilde{D}_t : t \geq 0)$ denote a new process which results when all of the uploads of piece one signed by a root peer (in the comparison system) are counted at the arrival time of the root peer. Since $\widetilde{D}$ counts the same events as $\widehat{D}$, but does so earlier, $\widehat{D}_t \leq \widetilde{D}_t$ for all $t \geq 0$. It is sufficient to prove (3.11) with D replaced by $\widetilde{D}$.

The random process $\widetilde{D}$ is a compound Poisson process. Jumps occur at the arrival times of root peers, which form a Poisson process of rate ξU. Let J be the size of the jump of $\widetilde{D}$ associated with a typical root peer. Then $J = J_1 + J_2$, where (a) J_1 is the number of infected peers holding piece one signed by the root peer, not counting the root peer itself, and (b) J_2 is the number of uploads of piece one to the one-club by the root peer and these J_1 peers. Consider one $M/GI/1$ queueing system with arrival rate $\xi\mu$ and service times following the distribution of a random variable $\widetilde{X} \sim Gamma(K-1, 2/\mu)$. Then the sum of all the times that the root peer and these J_1 peers are in the comparison system is the same as the duration, L, of a busy period of the $M/GI/1$ queue. And J_1 has the same distribution as the number of customers in a busy period of the $M/GI/1$ queue, not counting the customer who started the busy period. Note that ρ in (3.3) is the load factor for the $M/GI/1$ queue: $\rho = \xi\mu E[\widetilde{X}]$. Apply Lemma A.5.1 and Assumption 3.6.1, and following a similar argument as in [12, Page 259], we have $E[J] = \frac{1+\mu E[\widetilde{X}]}{1-\rho} - 1 \leq 4K$, and $E[J^2] \leq 2\{E[J_1^2] + E[J_2^2]\} \leq 64K^2$. Lemma 3.6.5 follows.

C.3 Proof of Lemma 3.7.3

We compare $Q(W)$ and $\mathcal{Q}W$ term by term. Consider terms $Q(T_C)$ and $\mathcal{Q}T_C$. First, assume $|C| \leq K-2$. Because α is fixed, we have $|Q(T_C) - \mathcal{Q}T_C| \leq a_1 + \alpha(a_2 + a_3)$,

$$
\begin{aligned}
a_1 &= \left|\frac{1}{2}Q(E_C^2) - E_C Q(E_C)\right| \leq \lambda_{total} + D_{total},\\
a_2 &= \left|Q(E_C\phi(H_C)) - Q(E_C)\phi(H_C) - E_C Q(\phi(H_C))\right|,\\
a_3 &= |Q(E_C)\phi(H_C)| \leq M_\phi(\lambda_{total} + D_{total}).
\end{aligned}
$$

The only way E_C and $\phi(H_C)$ can simultaneously change is that some peer with type in $\mathcal{E}_C$ becomes a peer with type in $\mathcal{H}_C$, causing E_C to decrease

by one, and $\phi(H_C)$ to decrease by at most one, so $a_2 \leq \Gamma_{\mathcal{E}_C,\mathcal{H}_C}$. Notice the fact that $\Gamma_{\mathcal{E}_C,\mathcal{H}_C} \leq D_{total}$, we have $\forall |C| \leq K-2$, $|Q(T_C) - \mathcal{Q}T_C| \leq M_\phi(D_{total}+1)\Theta(1)$. Secondly, assume $|C| = K-1$. Then, $|Q(T_C) - \mathcal{Q}T_C| = a_1 \leq \lambda_{total} + D_{total} \leq M_\phi(D_{total}+1)\Theta(1)$. There are only finitely many terms of T_C in W (2^K in total), and notice that $r < 1$, Lemma 3.7.3 follows.

C.4 Proof of Lemma 3.7.4

The upper bound for the drift of E_C is obvious: $Q(E_C) \leq \lambda_{total} = \Theta(1)$. Next consider $Q(\phi(H_C))$. Because ϕ is a decreasing function, only the rate for H_C to decrease contributes to the positive part in the drift of $\phi(H_C)$, so to consider an upper bound of $Q(\phi(H_C))$ it suffices to consider the rates of transitions which reduce H_C. There is only one way for H_C to decrease: peers with types in $\mathcal{H}_C$ to download a novel piece – with aggregate rate $D_{\mathcal{H}_C}$. Each peer with type in $\mathcal{H}_C$ downloading a novel piece can cause $\phi(H_C)$ to increase at most one. Thus, an upper bound for the drift of $\phi(H_C)$ is $Q(\phi(H_C)) \leq D_{\mathcal{H}_C} \leq U + H_C\mu = \Theta(1) + H_C\Theta(1)$.

We can choose d large enough, i.e. $d > 1$, so $M_\phi > 2d + 1/\beta + 1$. Thus $Q(\phi(H_C))$ vanishes when $H_C > M_\phi$, because $\phi(H_C)$ vanishes when $H_C > 2d + 1/\beta$ and the decreasing of H_C when state changes is bounded by $1 < d$. Hence $Q(\phi(H_C)) \leq M_\phi\Theta(1)$, because $M_\phi > 1$.

Finally, the bound on $\mathcal{Q}T_C$ follows from the other two bounds already proved.

C.5 Proof of Lemma 3.7.5

Assume that $n_S/n > 1 - \sigma$, where $0 < \sigma < \frac{1}{4}$ is to be specified. We consider two cases. Suppose first $|S| = K-1$. Peers with cache S only miss one piece $i \in \mathcal{F} \setminus S$, and $\mathcal{Q}T_S = E_S Q(E_S) \leq E_S\big[\sum_{C:i\in C} \lambda_C - U(1-\sigma)\big] \leq -\frac{1}{2}\Delta E_S$, if σ is set to be small enough: $\sigma < \Delta/(2U)$. Lemma 3.7.5 follows.

Suppose now that $|S| \leq K-2$, then by Lemma A.2.1,

$$\begin{aligned} Q(\phi(H_S)) &\leq \phi'(H_S)Q(H_S) + \beta/2(A_{\mathcal{H}_S} + D_{\mathcal{H}_S}) \\ &= \phi'(H_S)(A_{\mathcal{H}_S} - D_{\mathcal{H}_S}) + \beta/2(A_{\mathcal{H}_S} + D_{\mathcal{H}_S}). \end{aligned}$$

Substituting the above inequality into $Q(E_S) + \alpha Q(\phi(H_S))$, which is one component in Definition 3.7.12, and applying Lemma 3.7.2,

$$Q(E_S) + \alpha Q(\phi(H_S)) \leq \bar{\varsigma} - \varpi, \text{ where} \tag{C.1}$$
$$\begin{cases} \bar{\varsigma} := \lambda_{total} - \frac{1}{2} D_S + \alpha \phi'(H_S) A_{\mathcal{H}_S} \\ \varpi := \frac{1}{2} D_S + \alpha \phi'(H_S) D_{\mathcal{H}_S} - \alpha\beta/2 (A_{\mathcal{H}_S} + D_{\mathcal{H}_S}) \end{cases}$$

We claim first that $\bar{\varsigma} \leq -\Delta/2$. To prove this, suppose first that $H_S < d$: Then, $\phi'(H_S) = -1$ and because the seed applies RN, $A_{\mathcal{H}_S} \geq \Gamma_{\mathcal{P}_S, \mathcal{H}_S} \geq U(1-\sigma)/K \geq U/(4K)$. So $\bar{\varsigma} \leq \lambda_{total} - \alpha U/(4K) \leq -\Delta/2$. Suppose secondly that $H_S \geq d$: Then, $D_S \geq d\mu(1-\sigma) \geq \frac{1}{2} d\mu$. So $\bar{\varsigma} \leq \lambda_{total} - \frac{1}{2} d\mu \leq -\frac{1}{2}\Delta$, for d large enough: $d > 2(\lambda_{total} + \Delta/2)/\mu$.

We further claim that $\varpi \geq 0$. To prove this, let ω_S be the number of peers holding pieces not in S. For $\sigma < \frac{1}{2}$, we have

$$D_S \geq (U + \omega_S \mu)(1-\sigma) \geq \frac{1}{2}(U + \omega_S \mu). \tag{C.2}$$

Notice that pieces novel to peers with types in $\mathcal{H}_S$ are not contained in S. The number of peers that can upload pieces to peers with types in $\mathcal{H}_S$ is no larger than ω_S, so

$$D_{\mathcal{H}_S} \leq (U + \omega_S \mu)\sigma \leq 2\sigma D_S \tag{C.3}$$

In addition, $A_{\mathcal{H}_S} = \Gamma_{\mathcal{P}_S, \mathcal{H}_S} + \Gamma_{I_S, \mathcal{H}_S}$, where $I_S := \{\langle C', S' \rangle : |S'| = |S|, S' \neq S\}$. The number of peers with types in I_S is no larger than ω_S. Therefore $\Gamma_{I_S, \mathcal{H}_S} \leq U + \omega_S \mu \leq 2D_S$ by (C.2). And noticing that $\Gamma_{\mathcal{P}_S, \mathcal{H}_S} \leq D_S$, we have $A_{\mathcal{H}_S} \leq 3D_S$. Combining it with Lemma 3.7.2, and (C.3), we have $\varpi \geq \frac{1}{2} D_S - 2\alpha D_{\mathcal{H}_S} - \frac{\alpha\beta}{2} A_{\mathcal{H}_S} \geq (\frac{1}{2} - 4\alpha\sigma - \frac{3}{2}\alpha\beta) D_S$. We can set $4\alpha\sigma < \frac{1}{4}$, $\frac{3}{2}\alpha\beta < \frac{1}{4}$ so that $\varpi \geq 0$ follows.

Therefore, $\bar{\varsigma} \leq -\frac{1}{2}\Delta$ and $\varpi \geq 0$ imply that, when $n_S/n > 1 - \sigma$ and $|S| \leq K - 2$, $\mathcal{Q}T_S = [Q(E_S) + \alpha Q(\phi(H_S))] E_S \leq -\frac{1}{2}\Delta E_S$ and Lemma 3.7.5 follows.

C.6 Proof of Lemma 3.7.6

First, consider Lemma 3.7.6(a). Since there are only finitely many types, we can fix a set $S \subsetneq \mathcal{F}$ and consider the set of Class I states $\mathbf{n}$ for which $n_S/n > 1-\sigma$. Since $\sigma \in (0, \frac{1}{2})$, $E_S > \frac{1}{2}n$. By Definition 3.7.1, $\triangle > 0$. By Lemma 3.7.5,

$$\mathcal{Q}T_S \leq -\frac{1}{4}\triangle n = -\Theta(n). \tag{C.4}$$

Consider two conditions: (a) for type C with $|C| > |S|$, E_C may be larger than E_S. Lemma 3.7.4 and (C.4) imply $r^{|C|}\mathcal{Q}T_C \leq rM_\phi r^{|S|}\Theta(n) < 2^{-K-1}r^{|S|}\left|\mathcal{Q}T_S\right|$, if rM_ϕ is selected to be small enough; (b) for type C with $|C| \leq |S|$ but $C \neq S$, $E_C \leq \sigma n$. Note that $r^{|C|}r^K \leq r^{|S|}$, Lemma 3.7.4 and (C.4) imply $r^{|C|}\mathcal{Q}T_C \leq r^{|C|}M_\phi\Theta(E_C) \leq \sigma M_\phi r^{-K}r^{|S|}\Theta(n) < 2^{-K-1}r^{|S|}\left|\mathcal{Q}T_S\right|$, if $\sigma M_\phi r^{-K}$ is selected to be small enough.

Noticing that $E_S \geq n/2$, and applying Lemma 3.7.5, we have $\mathcal{Q}W = r^{|S|}\mathcal{Q}T_S + \sum_{C:C\neq S} r^{|C|}\mathcal{Q}T_C \leq r^{|S|}\mathcal{Q}T_S + \frac{1}{2}r^{|S|}\left|\mathcal{Q}T_S\right| \leq -\frac{1}{8}r^{|S|}\triangle n \leq -r^K\Theta(n)$, which proves Lemma 3.7.6(a).

Second, consider Lemma 3.7.6(b). Suppose $C_1 \not\subseteq C_2$ and consider the set of Class II states $\mathbf{n}$ such that $n_{C_1}/n > \eta, n_{C_2}/n > \eta$, where $\eta = \sigma/2^K$. In such states: $D_{C_2} \geq n_{C_2}n_{C_1}\mu/n \geq \mu\eta^2 n \in \sigma^2\Theta(n)$. Notice that $E_{C_2} \geq n_{C_2} \geq \eta n$, we have $E_{C_2}Q(E_{C_2}) \leq E_{C_2}(\lambda_{\mathcal{E}_{C_2}} - D_{C_2}) \leq -\sigma^3\Theta(n^2) + \Theta(n)$. Lemma 3.7.4 indicates $E_{C_2}Q(\phi(H_{C_2})) \leq M_\phi\Theta(n)$, so

$$\mathcal{Q}T_{C_2} = E_{C_2}Q(E_{C_2}) + \alpha E_{C_2}Q(\phi(H_{C_2})) \leq -\sigma^3\Theta(n^2) + M_\phi\Theta(n).$$

Obviously the above inequality works for the case $|C_2| = K-1$ where the ϕ term does not exist too. Applying the above inequalities and the result $\forall C$, $\mathcal{Q}T_C \leq M_\phi\Theta(n)$ from Lemma 3.7.4, we claim that, over the set of all Class II states, $\mathcal{Q}W \leq r^{|C_2|}\mathcal{Q}T_{C_2} + \sum_{C:C\neq C_2}\mathcal{Q}T_C \leq -r^K\sigma^3\Theta(n^2) + M_\phi\Theta(n)$, which proves Lemma 3.7.6(b).

References

[1] B. Cohen, "Incentives build robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[2] Wikipedia, "Bittorrent — Wikipedia, the free encyclopedia," 2013. [Online]. Available: http://en.wikipedia.org/wiki/BitTorrent#cite_note-1

[3] L. Massoulié and M. Vojnović, "Coupon replication systems," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 603–616, 2008.

[4] X. Zhou, S. Ioannidis, and L. Masosulié, "On the stability and optimality of universal swarms," in *ACM SIGMETRICS*, 2011.

[5] S. Sanghavi, B. Hajek, and L. Massoulie, "Gossiping with multiple messages," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4640–4654, 2007.

[6] B. Oguz, V. Anantharam, and I. Norros, "Stable, scalable, decentralized P2P file sharing with non-altruistic peers," *CoRR*, vol. abs/1107.3166, 2011.

[7] J. Zhu and B. Hajek, "Stability of a peer-to-peer communication system," *IEEE Transactions on Information Theory*, vol. 58, no. 7, pp. 4693–4713, 2012.

[8] J. Zhu, S. Ioannidis, N. Hegde, and L. Massoulie, "Stable and scalable universal swarms," in *ACM PODC*, 2013.

[9] X. Yang and G. de Veciana, "Service capacity of peer to peer networks," in *IEEE INFOCOM*, 2004.

[10] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *ACM SIGCOMM*, 2004.

[11] D. S. Menasche, A. A. Rocha, B. Li, D. Towsley, and A. Venkataramani, "Content availability and bundling in swarming systems," in *ACM CoNEXT*, 2009.

[12] B. Hajek and J. Zhu, "The missing piece syndrome in peer-to-peer communication," *Stochastic Systems*, vol. 1, pp. 246–273, 2011.

[13] S. Tewari and L. Kleinrock, "Proportional replication in peer-to-peer networks," in *IEEE INFOCOM*, 2006.

[14] J. Mundinger, R. Weber, and G. Weiss, "Optimal scheduling of peer-to-peer file dissemination," *Journal of Scheduling*, vol. 11, no. 2, pp. 105–120, 2008.

[15] D. S. Menasché, L. Massoulié, and D. Towsley, "Reciprocity and barter in peer-to-peer systems," in *IEEE INFOCOM*, 2010.

[16] B. Oğuz, V. Anantharam, and I. Norros, "Stable, distributed p2p protocols based on random peer sampling," in *IEEE Allerton*, 2012, pp. 915–919.

[17] Y. Zhou, D. Chiu, and J. Lui, "A simple model for analyzing P2P streaming protocols," in *IEEE ICNP*, 2007.

[18] I. Norros, H. Reittu, and T. Eirola, "On the stability of two-chunk file-sharing systems," *Queueing Systems*, vol. 67, no. 3, pp. 183–206, 2011.

[19] J. Han, T. Chung, S. Kim, T. T. Kwon, H.-c. Kim, and Y. Choi, "How prevalent is content bundling in bittorrent," in *ACM SIGMETRICS*, 2011.

[20] D. Wu, C. Liang, Y. Liu, and K. Ross, "View-upload decoupling: A redesign of multi-channel P2P video systems," in *IEEE INFOCOM*, 2009.

[21] H. Schulze and K. Mochalski, "Internet study 2008/2009," *IPOQUE Report*, vol. 37, pp. 351–362, 2009.

[22] X. Zhang, J. Liu, B. Li, and Y. Yum, "Coolstreaming/Donet: a data-driven overlay network for peer-to-peer live media streaming," in *IEEE INFOCOM*, 2005.

[23] Z. Liu, C. Wu, B. Li, and S. Zhao, "Uusee: large-scale operational on-demand streaming with random network coding," in *IEEE INFOCOM*, 2010.

[24] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang, "Challenges, design and analysis of a large-scale P2P-vod system," in *ACM SIGCOMM*, 2008.

[25] D. Wu, Y. Liu, and K. W. Ross, "Queuing network models for multi-channel P2P live streaming systems," in *IEEE INFOCOM*, 2009.

[26] N. Magharei and R. Rejaie, "Prime: Peer-to-peer receiver-driven mesh-based streaming," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1052–1065, 2009.

[27] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *ACM SIGOPS Operating Systems Review*, 2003.

[28] J. Kim and R. Srikant, "Peer-to-peer streaming over dynamic random Hamilton cycles," in *IEEE Information Theory and Applications Workshop*, 2012.

[29] D. Tomozei and L. Massoulié, "Flow control for cost-efficient peer-to-peer streaming," in *IEEE INFOCOM*, 2010.

[30] A. T. Nguyen, B. Li, and F. Eliassen, "Chameleon: Adaptive peer-to-peer streaming with network coding," in *IEEE INFOCOM*, 2010.

[31] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-to-peer streaming," in *IEEE ICNP*, 2003.

[32] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," in *ACM SIGOPS Operating Systems Review*, 2003.

[33] D. A. Tran, K. A. Hua, and T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," in *IEEE INFOCOM*, 2003.

[34] W. Zhang, Q. Zheng, H. Li, and F. Tian, "An overlay multicast protocol for live streaming and delay-guaranteed interactive media," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 20–28, 2012.

[35] E. Brosh and Y. Shavitt, "Approximation and heuristic algorithms for minimum delay application-layer multicast trees," in *IEEE INFOCOM*, 2004.

[36] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Trans. Program. Lang. Syst.*, vol. 5, no. 1, pp. 66–77, 1983.

[37] K. Mokhtarian and H.-A. Jacobsen, "Minimum-delay overlay multicast," in *IEEE INFOCOM*, 2013.

[38] S. Y. Shi and J. S. Turner, "Routing in overlay multicast networks," in *IEEE INFOCOM*, 2002.

[39] J. Widmer, A. Capalbo, A. F. Anta, and A. Banchs, "Rate allocation for layered multicast streaming with inter-layer network coding," in *IEEE INFOCOM*, 2012.

[40] V. K. Goyal, "Multiple description coding: Compression meets the network," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 74–93, 2001.

[41] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[42] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *IEEE INFOCOM*, 2005.

[43] J. Zhu and B. Hajek, "Tree dynamics for peer-to-peer streaming," *ArXiv e-prints*, 2013.

[44] J. Kim and R. Srikant, "Achieving the maximum P2P streaming rate using a small number of trees," in *ICCCN*, 2011.

[45] R. Kumar, Y. Liu, and K. Ross, "Stochastic fluid theory for P2P streaming systems," in *IEEE INFOCOM*, 2007.

[46] Y. Zhou, D.-M. Chiu, and J. Lui, "A simple model for chunk-scheduling strategies in P2P streaming," *IEEE/ACM Transactions on Networking*, vol. 19, no. 1, pp. 42–54, 2011.

[47] L. Ying, R. Srikant, and S. Shakkottai, "The asymptotic behavior of minimum buffer size requirements in large P2P streaming networks," in *Information Theory and Applications Workshop*, 2010.

[48] J. Almeida, D. Eager, M. Vernon, and S. Wright, "Minimizing delivery cost in scalable streaming content distribution systems," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 356–365, 2004.

[49] Y. Lu, J. Mol, F. Kuipers, and P. Van Mieghem, "Analytical model for mesh-based p2pvod," in *IEEE International Symposium on Multimedia*, 2008.

[50] Y. Boufkhad, F. Mathieu, F. De Montgolfier, D. Perino, and L. Viennot, "Achievable catalog size in peer-to-peer video-on-demand systems," in *International conference on Peer-to-peer systems*, 2008.

[51] Y. Zhou, T. Z. J. Fu, and D. M. Chiu, "On replication algorithm in P2P VoD," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, p. 1, 2012.

[52] B. Tan and L. Massoulié, "Optimal content placement for peer-to-peer video-on-demand systems," in *IEEE INFOCOM*, 2011.

[53] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello, "Push-to-peer video-on-demand system: Design and evaluation," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1706–1716, 2007.

[54] B. Hajek, "An exploration of random processes for engineers," December 20, 2011. [Online]. Available: http://www.ifp.illinois.edu/~hajek/Papers/randomprocesses.html

[55] L. Leskelä, P. Robert, and F. Simatos, "Interacting branching processes and linear file-sharing networks," *Advances in Applied Probability*, vol. 42, no. 3, pp. 834–854, 2010.

[56] K. B. Athreya and S. S. N. Lahiri, *Measure theory and probability theory*. Springer, 2006.

[57] F. Foster, "On the stochastic matrices associated with certain queuing processes," *The Annals of Mathematical Statistics*, vol. 24, no. 3, pp. 355–360, 1953.

[58] S. Deb, M. Médard, and C. Choute, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2486–2507, 2006.

[59] S. P. Meyn and R. L. Tweedie, *Markov chains and stochastic stability*. Cambridge University Press, 2009.

[60] J. Kingman, "Some inequalities for the queue gi/g/1," *Biometrika*, vol. 49, no. 3/4, pp. 315–324, 1962.